NAV

Introduction

Proton Base URL

```
https://[sandbox][api].hydrogenplatform.com/proton/v1
```

All client facing no-code apps utilize Proton APIs in a pre-approved, compliant manner, but these APIs can be used standalone in select enterprise use-cases.

The Hydrogen Proton API is designed for added business logic on top of the core Nucleus data model within Hydrogen. Proton offers analytical tools, scoring, recommendations, and simulations.

Run in Postman

Authentication

All Hydrogen APIs use the same OAuth 2.0 authorization. Please refer to the Nucleus API for detailed documentation.

Client Credentials

If you are utilizing the Integration client libraries for `client_credentials` authorization, please view the sample code in the language of your choice in the right panel.

Resource Owner Password Credentials

NAV

If you are utilizing the client libraries for `password` authorization, please view the sample code in the language of your choice in the right panel.

Fields

IDs

IDs are represented in universally unique identifier (UUID) format. A UUID is a string of 32 alphanumeric characters in the format 8-4-4-4-12. An example would be `efa289b2-3565-42e6-850b-8dad25727e99`. All requests that reference Nucleus data entities must utilize this UUID format.

DATES

Dates are represented in ISO-8601 format, as YYYY-MM-DD. An example would be `2018-01-10`.

Errors

ERROR CODES

Code	Description
`400`	*Bad Request.*
`401`	*Unauthorized.* Occurs when you are using an invalid or expired access token.
`403`	*Forbidden.* The request was valid but you are not authorized to access the resource.
`404`	*Not Found.* Occurs when you are requesting a resource which doesn't exist such as an incorrect URL or incorrect ID.
`429`	*Too Many Requests.* Exceeded the rate limit set. Currently, there is no rate limit on the APIs.
`500`	*Internal Server Error.*
`503`	*Service Unavailable.* If the API is down for maintenance you will see this error.

NAV

STATUS CODES

Code	Description
`200`	*Ok.* The request was successful.

Versioning

The Proton API is currently in major version 1.0. All features which are not backwards compatible will be pushed as a major version release. Features that we consider to be backwards compatible include the following:

- Adding new API resources
- Adding new optional request parameters to existing API methods
- Adding new fields to existing API responses
- Changing the order of fields in existing API responses
- Changing the length or format of objects

Changelog

Changes and updated versions will be outlined here.

Date	Change	Description
2021-06-07	`update`	Enabled support for household-level and business-level analytics to various services. Introduced a new `transaction_status_scope` parameter to improve analytics granularity for transaction-based services. Expanded functionality for Card Authorization. Improved periodicity handling in Budget Calculator. Enabled new functionality in Recurring Transaction Analysis to flag transaction records that are identified as recurring. Improved error handling for the Cards module. Fixed various divide-by-zero errors. Fixed a bug in Cash Flow Analysis that led to incorrect transactions being included in the analysis for certain requests.
2021-06-07	`addition`	Added the Spending Analysis service to the Personal Financial Management module. Added the Card Analysis service to the Cards module. Added the Performance Calculator and Decision Tree Result services to the Wealth module. Added the Order Rebalance service to the Portfolio Management module.

NAV

Date	Change	Description
2020-12-31	`addition`	Added a Cards module with two new services: Card Authorization and Card Limit Check. Added support for `currency_conversion` across various services. Also added Integrations support for services that consume market data via the `market_data_source` parameter. Budget Calculator and Cash Flow Analysis now support both held and held-away account data via the `scope` parameter. Added functionality to Budget Calculator to calculate average historical spending across budget categories. Added an `only_cleansed` parameter to services that consume Portfolio Transactions. Standardized the handling of boolean input parameters to minimize ambiguity. Fixed several bugs in the processing of proxy security data across applicable services.
2020-09-30	`addition`	Added two services to the Personal Financial Management module: Fee Analysis and Recurring Transaction Analysis.
2020-03-04	`update`	Relaxed constraints on underlying data in Financial Picture by treating a lack of balance records with non-zero balances as 0 net worth. Enhanced error messaging for Goal Accumulation Allocation when underlying security price data is insufficient. Added more descriptive error messaging when a Proton request fails due to an issue with another Hydrogen product such as Nucleus API or Integration API. Improved input validation on the `opt_config` object and more descriptive error messaging when passing it empty strings as tickers for the following endpoints: Goal Accumulation Allocation, Goal Decumulation Allocation, Risk Allocation, Portfolio Optimization Score. Updates to Budget Calculator: Changed `as_of_date` to use UTC instead of the local current date. Fixed issues where `aggregation_accounts` was not properly considered and where spending was not accurately reflected on budgets with a `frequency_unit` of `daily`. Relaxed constraints on underlying data by treating a lack of valid transaction records for a budget as 0 spending. Enhanced results for a budget with a `null` `subcategory` so that it will consider all transactions that map at the `category` level. Improved error handling when passing an invalid `budget_id`.
2020-02-14	`update`	Made `initial_balance` parameter optional with a default value of 0 for the following endpoints: Annuity Calculator, Education Calculator, Savings Calculator, Variable Annuity. Nucleus entity connected to Goal Accumulation Allocation and Goal Decumulation Allocation endpoints. Updated Risk Allocation to make the `risk_score` parameter optional with a default value of 50, enhance input validation and error messaging, and fix a bug that prevented the 0-100 range on this parameter from being enforced. Upgraded the proxy data framework when `use_proxy_data` is `true` for various services. Improved error messaging for Goals and Risk Scoring when sourcing data from Nucleus, and for Goal Decumulation when there is an invalid periodicity relationship between `withdrawal_config` amounts and the goal horizon. Fixed a series of bugs:

NAV

Date	Change	Description
		Allowed a `result_type` of `custom` to be accurately reflected in the results for Variable Annuity. Updated Risk Allocation to prevent the `allocations` input from being enforced as conditionally required. Adjusted Life Insurance Needs Calculator to allow optional inputs to have their default values properly set. Changed Goal Accumulation services to prevent certain nested inputs in the `recommendation_config` field from being required and to properly process `deposit_config` period references for certain combinations of deposit frequency and goal horizon frequency. Changed Goal Decumulation services to produce the proper error message when `d_horizon` has a value of 0 and to allow for successful analysis in some cases when `a_horizon` has a value of 0.
2019-12-19	`update`	Nucleus data entities connected to Goals, Simulations, and Financial Planning. Audit log functionality added to Goals and Simulations using the parameter `create_log` in request parameter.
2019-08-22	`update`	Fixed a series of bugs: Enabled Financial Health Check to properly handle cases where the ratio denominator is 0. Updated Sensitivity Analysis & Scenario Analysis to manually override `t-stat` to `None/null` when it comes back as `NaN`. Updated Education Calculator & Retirement Calculator to prevent excess cash to be left in `deposit_schedule` at the end of the time horizon when an inflation rate was applied. Changed the order of fields in `return_details` within Annuity Calculator to be consistent with other endpoints, additionally, fixed a bug that sometimes prevented a successful analysis. Changed `lag` parameter in Sensitivity Analysis to be optional and default to 0. Finally, updated select error responses to be in proper JSON format.
2019-08-22	`update`	Updated Emergency Fund Calculator to remove a restriction from the `interest_rate` parameter. Updated Savings Calculator to remove the constraint on the length of the `return_schedule` array. Added a series of new parameters to the Monte Carlo simulation. Added `use_proxy_data` to the following endpoints Mean-Variance Optimization, Event Study, Sensitivity Analysis, Scenario Analysis, Portfolio What If, Diversification Score, Portfolio Optimization Score, Goal Decumulation, Goal Accumulation, Risk Allocation, Variable Annuity. Added new response fields to the Financial Planning and Annuity Calculators to better summarize the output.
2019-08-22	`addition`	Added three new services to the Personal Financial Management module: Budget Calculator, Cash Flow Analysis, and Financial Picture. The Cash Flow Analysis Tool displays trends related to the users spending and cash flow, allowing them to determine where they can cut down their spending and save money. The Financial Picture Tool runs analytics on all held-away accounts that have been aggregated so that the user may see a total picture of their financial life. The Budget Calculator

NAV

Date	Change	Description
		pulls in the user's spending and budget information from Nucleus, which is then used to generate insightful information about their budget by category and merchant. Finally, we added an additional service to the Annuities module: Variable Annuity; this service helps to evaluate potential outcomes and get a better understanding of how a variable annuity product may behave.
2019-03-26	`update`	Fixed a bug in which certain Backtest requests would fail based on fluctuating model securities. Fixed a bug in which the impact of inflation was reflected inconsistently across Financial Planning calculators. Fixed a bug in which rates of return were incorrectly annualized in Portfolio Optimization Score. Fixed a bug in which the Mean-Variance Optimization service would not return frontier portfolios when one frontier point encountered an error. Fixed a bug in which Mean-Variance Optimization would fail based on the absence of optional security asset class data. Fixed a bug in which the arithmetic mean return was used instead of the geometric mean in Goals allocation services.
2019-03-26	`update`	Introduced two new parameters to all services in the Goals module, `adjust_for_compounding` and `compounding_rate`, which help to approximate a compounding effect on deposits for specific request configurations.
2019-03-26	`addition`	Added three new services to the Simulations module: Sensitivity Analysis, Scenario Analysis, and Portfolio What-If. These tools offer the ability to better understand an investment portfolio by simulating its behavior under a variety of circumstances. The former two services simulate the impact of external changes, while the latter considers changes within the portfolio itself.
2019-01-24	`update`	Fixed a bug in the Goals module in which portfolio risk was improperly calculated when converting between certain time frequencies. Added handling for Mortgage Calculator cases in which `down_payment` could be negative.
2018-12-07	`update`	Included new response fields for the Backtest service to summarize the risk and return of the backtested model.
2018-12-07	`addition`	Added the Event Study service to the Simulations module, which simulates a portfolio's behavior during key historical events. Added the Financial Health Check service to the Financial Health module, which offers the ability to assess an individual's status through a series of financial ratios.
2018-11-12	`addition`	Added two services to the Financial Health module: Portfolio Optimization Score, which evaluates a portfolio against its optimal counterpart, and Diversification Score, which assesses the level of diversification found in an investor's portfolio.

NAV

Date	Change	Description
		Added an Annuity module with five Annuity Calculator services for the analysis of fixed annuity streams.
2018-08-08	`update`	Enhanced all services in the Goals module, making the framework more robust by allowing for greater levels of customization and the ability to handle a wider array of goal configurations.
2018-07-02	`addition`	Added a suite of financial calculators, including Education, Mortgage, Life Insurance Needs, Major Purchase, Retirement, and Savings.

Terminology

Throughout this documentation, there will be a number of financial terms and concepts that may be confusing to the developer not privy to financial jargon. After first outlining basic platform terms, we have defined some financial terms that may be referred to or represented in the parameters or their descriptions.

Nucleus Terms

Name	Description
Client	A client is identified based on a unique identifier, usually a government issued ID (e.g. social security number). A client can have multiple cards, accounts, aggregation accounts, budgets, models, allocations and goals.
Household	A household is a grouping of clients in the same family, which can be used to view their finances aggregated across the household.
Business	A business represents a corporate entity where collections of clients are employed. A client may be designated as the owner of a business.
Contact	A contact represents a company that purchases goods or services from a business (customer), or a company that supplies good or services to a business (supplier).

NAV

Name	Description
Invoice	An invoice represents an account receivable owed by a customer to a business (invoice), or an account payable owed by a business to a supplier (bill).
Financial Statement	A financial statement represents statistics about a business generated via financials such as an income statement and balance sheet.
Card	A card can be issued to a client via a card program manager.
Card Program	A card program is issued and run by a program manager who is responsible for establishing relationships with processors, sponsor banks, and card networks.
Spending Control	A spending control can be added at a client level for all cards associated with the client. These controls will restrict purchases and ATM withdrawal amounts, as well as spending within a category or at a specific merchant or location.
Aggregation Account	A client can have one or more aggregation accounts. These accounts are external, meaning they are held-away at various other institutions, and they may have balances, transactions, and/or holdings associated with them.
Budget	A budget allows a client to track their spending on a card, account, or aggregation account(s) against their budget over time.
Goal	A desired financial outcome in the future, defined by time and monetary value.
Allocation	An abstract collection of one or more models grouped together and strategically weighted. Each account is assigned to one allocation, which can contain multiple models.
Model	An abstract collection of one or more securities grouped together and strategically weighted. An account can be subscribed to n-number of models through an allocation. Each model holds n-number of securities and may be a component of n-number of allocations.
Account	A client can have one or more accounts and each account can be assigned to one allocation. An account can hold n-number of portfolios.
Portfolio	Portfolios are unique to a given account, while models have holdings with theoretical strategic weights; portfolios reflect actual account holdings, which depend on changes in the model, account additions and withdrawals, dividends, canceled trades, and other account-specific activity. Portfolios may or may not be subscribed to a model, which can be a component of an allocation.

NAV

Name	Description
Security	A security is defined as any financial instrument or product that an investor may own. Securities have certain attributes, like prices and unique identifiers. Securities are held within models and portfolios.
Asset Size	The aggregate monetary value of an investment account based on underlying positions. The asset size changes as securities fluctuate or monies/securities are deposited/withdrawn.

Investing Terms

Name	Description
Accumulation	Accumulation goals are defined by a target amount of capital in the future, assumed to be redeemed in full upon termination of the goal horizon.
Confidence Target	Confidence targets, indicated by `conf_tgt` when applicable, serve as the basis for a percentile analysis of Monte Carlo outcomes, in which outcomes below the ((1 – c)*100)th percentile, where c is the confidence target on a scale from 0 to 1, are excluded as a left-tail margin. If the confidence target is not satisfied, this is reflected by a goal probability that is less than the confidence target.
Decumulation	Decumulation goals are defined by redeeming a target amount of capital periodically over a horizon, and may also include an accumulation phase.
Drift-Based Rebalancing	Drift-based rebalancing generates a signal whenever a portfolio holding deviates from its strategic target by a predefined amount.
Efficient Frontier	An efficient frontier is a collection of optimal portfolios across a maximal range of risk and return levels.
Goal Horizon	The amount of time until the funds are needed for a given goal. A goal may have a decumulation horizon as well as an accumulation horizon.
Monte Carlo	Using a random walk, this process estimates future values by simulating individual periods of portfolio growth based on samples derived from the given risk and return portfolio attributes, in conjunction with ongoing cash inflows or outflows (which occur at the end of each period) in the simulated account.

NAV

Name	Description
Optimization	A programmatic mathematical process by which securities are weighted to optimally achieve an objective.
Percentage	Throughout the API, percentages are represented in decimal format. For example, 0.02 translates to 2%.
Rebalance	Execute trades in an account in order to move holdings in line with the model or allocation strategic weights.
Tax (Principal and Non-Principal)	Principal_tax and nonprincipal_tax refer to an expected tax rate paid on the redemption of principal funds (e.g. contributed capital) and non-principal funds (e.g. gains/profits).
Ticker	The unique identifier of a given security. Unlike a CUSIP, this is a string of letters usually five characters or less.
Universe	The pool of securities that are acceptable to select for a given model, allocation or portfolio.

Cards

Card Analysis

Card Analysis provides a top-down view of card spending activity. Data may be analyzed at the client, business, card program, or tenant level. The analysis gives insight into top spenders, top cards, metrics such as average transaction size, and a historical view to see how card activity changes over time.

HTTP REQUEST

```
POST /card_analysis
```

Example Request

NAV

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "card_program_id": "48f67eea-0ae6-4396-add7-65f57c515367",
        "currency_code": "USD",
        "start_date": "2021-03-01",
        "end_date": "2021-05-11",
        "card_status_scope": [
            "activated"
        ],
        "transaction_status_scope": [
            "completed",
            "pending"
        ],
        "response_limit": 1,
        "history_frequency_interval": "month",
        "show_history": true,
        "show_top_cardholders": true,
        "show_top_cards": true
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/card_analysis"
```

ARGUMENTS

Parameter	Description
`client_id` *UUID*	The ID of a Nucleus Client to analyze. If provided, all cards linked to this client will be considered. If one of `client_id`, `business_id` or `card_program_id` is not passed, all clients within the tenant will be considered.
`business_id` *UUID*	The ID of a Nucleus Business to analyze. If provided, all clients linked to this business will be considered. If one of `client_id`, `business_id` or `card_program_id` is not passed, all clients within the tenant will be considered.
`card_program_id` *UUID*	The ID of a Nucleus Card Program to analyze. If provided, all clients linked to this card program will be considered. If one of `client_id`, `business_id` or `card_program_id` is not passed, all clients within the tenant will be considered.
`currency_code` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Only records with this currency code will be considered. Defaults to USD if `currency_conversion` is not provided. If

NAV

Parameter	Description
	`currency_conversion` is provided, all origin currencies will be considered by default. See Currencies.
`currency_conversion` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Records will be converted from their original currency to the one indicated here. Conversions are supported both to and from the following currency codes: `USD`, `GBP`, `EUR`, `AUD`, `CAD`, `CHF`. See Currencies.
`start_date` *date*	Start date of the analysis period. Defaults to the earliest date of available cardholder spending data.
`end_date` *date*	End date of the analysis period. Defaults to today's date.
`card_status_scope` *array[string]*	If populated, only Nucleus Cards whose status matches one of the array values are considered in the analysis. Defaults to `null`, meaning transaction data from all cards are considered regardless of the status.
`transaction_status_scope` *array[string]*	If populated, only Nucleus Portfolio Transactions whose status matches one of the array values are considered in the analysis. Defaults to `null`, meaning all transactions are considered regardless of the status.
`response_limit` *integer*	Maximum number of records to be returned in `top_businesses`, `top_cardholders`, and `top_cards`. Defaults to `10`.
`history_frequency_interval` *string*	The frequency of historical cardholder spending data. Value may be `day`, `week`, `month`, `quarter`, or `year`. Defaults to `month`.
`show_history` *boolean*	If `true`, returns a history of spending, in total, by business, by cardholder, and by card. Defaults to `false`.
`show_top_cardholders` *boolean*	If `true`, returns details on the top spending cardholders. Defaults to `false`.
`show_top_cards` *boolean*	If `true`, returns details on the top spending cards. Defaults to `false`.

NAV

Example Response


```
{
    "currency_code": "USD",
    "analysis_start": "2021-03-01",
    "analysis_end": "2021-05-11",
    "summary": {
        "number_of_cardholders": 5,
        "number_of_cards": 5,
        "total_amount_spent": 1925.0,
        "number_of_transactions": 6,
        "mean_transaction_amount": 320.83,
        "history": [
            {
                "period_start": "2021-03-01",
                "period_end": "2021-03-31",
                "amount_spent": 125.0
            },
            {
                "period_start": "2021-04-01",
                "period_end": "2021-04-30",
                "amount_spent": 1800.0
            },
            {
                "period_start": "2021-05-01",
                "period_end": "2021-05-11",
                "amount_spent": 0.0
            }
        ]
    },
    "top_cardholders": [
        {
            "name": "Crystal Jennings",
            "number_of_cards": 1,
            "total_amount_spent": 750.0,
            "number_of_transactions": 1,
            "mean_transaction_amount": 750.0,
            "history": [
                {
                    "period_start": "2021-03-01",
                    "period_end": "2021-03-31",
                    "amount_spent": 0.0
                },
                {
```

NAV


```
                    "period_start": "2021-04-01",
                    "period_end": "2021-04-30",
                    "amount_spent": 750.0
                },
                {
                    "period_start": "2021-05-01",
                    "period_end": "2021-05-11",
                    "amount_spent": 0.0
                }
            ]
        }
    ],
    "top_cards": [
        {
            "card_holder_name": "Crystal A Jennings",
            "institution_name": "Hydro Bank",
            "card_name": "Select Debit Card",
            "total_amount_spent": 750.0,
            "number_of_transactions": 1,
            "mean_transaction_amount": 750.0,
            "history": [
                {
                    "period_start": "2021-03-01",
                    "period_end": "2021-03-31",
                    "amount_spent": 0.0
                },
                {
                    "period_start": "2021-04-01",
                    "period_end": "2021-04-30",
                    "amount_spent": 750.0
                },
                {
                    "period_start": "2021-05-01",
                    "period_end": "2021-05-11",
                    "amount_spent": 0.0
                }
            ]
        }
    ]
}
```


RESPONSE

NAV

Field	Description
`currency_code`	Currency code associated with monetary response values.
`analysis_start`	Start date of the analysis period. If `start_date` was provided in the request, it is reflected here. If `start_date` was not provided, this value is derived dynamically.
`analysis_end`	End date of the analysis period. If `end_date` was provided in the request, it is reflected here. If `end_date` was not provided, this value is derived dynamically.
`summary`	High-level analysis of card spending.
`number_of_cardholders`	Number of cardholders, or Nucleus Clients with cards.
`number_of_cards`	Number of Nucleus Cards.
`total_amount_spent`	Total amount spent over the analysis period.
`number_of_transactions`	Number of card transactions, or Nucleus Portfolio Transactions linked to a card.
`mean_transaction_amount`	Average card transaction amount over the analysis period.
`history`	Historical total card spending data, with frequency dependent on the `history_frequency_interval` value passed in the request. Each entry has the fields shown below.
`period_start`	Period start date.
`period_end`	Period end date.
`amount_spent`	Amount spent in the period.
`top_cardholders`	Cardholders that spent the most over the analysis period. Each entry represents a single cardholder and has the fields shown below.
`name`	Name of the cardholder associated with the spending.
`number_of_cards`	Number of cards held by the cardholder.
`total_amount_spent`	Total amount spent over the analysis period.
`number_of_transactions`	Number of card transactions.

NAV

Field	Description
`mean_transaction_amount`	Average card transaction amount over the analysis period.
`history`	Historical cardholder spending data, with frequency dependent on the `history_frequency_interval` value passed in the request. Each entry has the fields shown below.
`period_start`	Period start date.
`period_end`	Period end date.
`amount_spent`	Amount spent in the period.
`top_cards`	Cards that spent the most over the analysis period. Each entry represents a single card and has the fields shown below.
`card_holder_name`	Name of the cardholder that appears on the card.
`institution_name`	Name of the institution that issued the card.
`card_name`	Name of the card associated with the spending.
`total_amount_spent`	Total amount spent over the analysis period.
`number_of_transactions`	Number of card transactions.
`mean_transaction_amount`	Average card transaction amount over the analysis period.
`history`	Historical card spending data, with frequency dependent on the `history_frequency_interval` value passed in the request. Each entry has the fields shown below.
`period_start`	Period start date.
`period_end`	Period end date.
`amount_spent`	Amount spent in the period.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Business`
- `Card`

NAV
- Card Program
- Account
- Portfolio
- Portfolio Transaction

Card Authorization

Card Authorization provides the ability to authorize a card transaction based on a variety of pre-defined spending controls. For spending control authorizations, information about the transaction, such as transaction type, transaction category, merchant, and location is consumed by the service. This information, along with the client's card transaction history, is compared to existing spending and transaction limits. If the limits have not been or will not be exceeded by the transaction, it will be authorized. Cleansing of raw transaction data, utilization of audit logs, and full and partial authorizations of the requested amount are supported.

HTTP REQUEST

```
POST /card_authorization
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "card_id": "3f2a4530-a0f5-4eca-a7b1-c151b1175f99",
        "currency_code": "USD",
        "auth_type": [
            "spending_control"
        ],
        "amount": 300,
        "date": "2020-11-20T08:00:00.000+0000",
        "transaction_type": "purchase",
        "merchant_id": "87ef9136-f7e1-4ef0-8dbc-e58bf435d11c",
        "merchant": "Delta",
        "merchant_category_code": 2345,
        "mid": "123456790",
        "transaction_category_id": "48f67eea-0ae6-4396-add7-65f57c515367",
        "transaction_category": "Travel",
        "description": "DELTA AIRLINES #235920",
        "memo":  null,
        "location": {
```

NAV

```
            "country": "US"
        },
        "partial_auth": false,
        "cleanse_data": false,
        "use_audit_log": false
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/card_authorization"
```

ARGUMENTS

Parameter	Description
`card_id` *UUID, required*	The ID of a Nucleus Card that the transaction corresponds to.
`currency_code` *string, required*	The alphabetic currency code of the transaction, limited to 3 characters. Only records with this currency code will be considered. See Currencies.
`auth_type` *array[string], required*	The type of authorization to be performed on the transaction. Value may be `spending_control`. If `spending_control` is passed, the transaction will be authorized if it does not exceed any of the existing Nucleus Spending Controls.
`amount` *float, required*	Amount of the transaction to be authorized. A negative value indicates a debit, while a positive value indicates a credit.
`date` *timestamp, required*	Date and time of the transaction to be authorized.
`transaction_type` *string, required*	Type of the transaction to be authorized. Value may be `purchase`, `atm_withdrawal`, or `other`.
`transaction_category` *string*	Name of the category of the transaction to be authorized. See Transaction Categories.
`transaction_category_id` *UUID*	ID of the category of the transaction to be authorized. See Transaction Categories.
`merchant` *string*	Name of the merchant of the transaction to be authorized. See Merchants.
`merchant_id` *UUID*	ID of the merchant of the transaction to be authorized. See Merchants.

NAV

Parameter	Description
`mid` *string*	Internal merchant ID of the transaction to be authorized.
`merchant_category_code` *string*	Merchant category code of the transaction to be authorized.
`description` *string*	Description of the transaction to be authorized.
`memo` *string*	Memo of the transaction to be authorized.
`location` *object*	Location details of the transaction to be authorized.
`country` *string*	ISO 3166 ALPHA-2 country code of the transaction to be authorized. See Countries.
`partial_auth` *boolean*	If `true`, a portion of the transaction `amount` may be authorized. Defaults to `false`, meaning only the full transaction `amount` may be authorized.
`cleanse_data` *boolean*	If `true`, some of the transaction data provided in this request will be submitted for data cleansing prior to any analysis. This data may include `merchant`, `merchant_category_code`, `description`, `memo`, and `amount`. Defaults to `false`, meaning that no additional data cleansing will be performed on the transaction data provided. This service requires access to the `plasma` premium offering.
`use_audit_log` *boolean*	If `true`, the most recent Audit Log created by Card Limit Check will be utilized to derive the current status of any spending controls. Defaults to `false`, meaning that the status of any spending controls will be derived dynamically in this analysis.

Example Response

```
{
    "is_authorized": true,
    "requested_amount": 1300.0,
    "authorized_amount": 1000.0,
```

NAV


```
        "rejected_amount": 300.0,
        "rejection_details": {
            "message": "atm_withdrawal_spending_limit",
            "spending_control_id": "5d6e3328-09f4-4450-bb56-2ab9751fc0d7",
            "control_type": "spending_limit",
            "control_scope": "atm_withdrawal",
            "control_value": null,
            "frequency_unit": "one_time",
            "frequency": 1,
            "limit_value": 1000.0,
            "limit_used": 0,
            "limit_remaining": 1000.0
        }
    }
```


RESPONSE

Field	Description
`is_authorized`	Indicates whether the transaction was successfully authorized. A value of `true` indicates that the transaction was fully or partially authorized, while a value of `false` indicates that the transaction was rejected.
`requested_amount`	Amount requested to be authorized.
`authorized_amount`	Amount authorized.
`rejected_amount`	Amount rejected.
`rejection_details`	Details of the rejection when all or a portion of the requested amount is rejected.
`message`	Message explaining the reason for the rejection. Value will be one of the following: `all_spending_limit`, `purchase_spending_limit`, `atm_withdrawal_spending_limit`, `transaction_category_spending_limit` `transaction_category_id_spending_limit`, `merchant_spending_limit`, `merchant_id_spending_limit`, `location_spending_limit`, `all_transaction_limit`, `purchase_transaction_limit`, `atm_withdrawal_transaction_limit`, `transaction_category_transaction_limit`, `transaction_category_id_transaction_limit`, `merchant_transaction_limit`, `merchant_id_transaction_limit`, `location_transaction_limit`, `transaction_category_allow_list`, `transaction_category_id_allow_list`, `merchant_allow_list`,

NAV

Field	Description
	`merchant_id_allow_list`, `location_allow_list`, `transaction_category_deny_list`, `transaction_category_id_deny_list`, `merchant_deny_list`, `merchant_id_deny_list`, `location_deny_list`, `mid_restriction`, `time_delay_restriction`.
`spending_control_id`	ID of the Nucleus Spending Control that triggered the rejection.
`control_type`	Type of the spending control. Value may be `spending_limit`, `transaction_limit`, `allow_list`, or `deny_list`.
`control_scope`	Scope of the spending control. Value may be `all`, `purchase`, `atm_withdrawal` or `merchant_category`.
`control_value`	Specific value that triggered the rejection. Will be one of the `control_values` from the corresponding spending control.
`frequency_unit`	Frequency unit of the spending control. Value may be `one_time`, `daily`, `weekly`, `monthly`, `quarterly`, or `annually`.
`frequency`	Frequency of the spending control. Indicates the number of `frequency_unit` between each period.
`limit_value`	Value of the spending control limit.
`limit_used`	Value already used towards this spending control limit.
`limit_remaining`	Value that remains to be used towards this spending control limit.
`existing_balance`	Card balance prior to the processing of any newly authorized transaction.
`new_balance`	Card balance after the processing of any newly authorized transaction.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Portfolio Transaction`
- `Transaction Code`
- `Card`
- `Spending Control`

NAV

Card Limit Check

Card Limit Check makes it easy to monitor usage of a client's cards relative to their set limits. This service analyzes a client's spending and transaction limits against their card transaction history for the specified period. A detailed status of each individual limit is returned, indicating how much of the limit has been used or is remaining, whether or not the limit has been breached, and how much time is left in the period before the limit resets.

HTTP REQUEST

```
POST /card_limit_check
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "client_id": "159bb162-b911-4c97-ba96-53493e62674d",
        "currency_code": "USD",
        "as_of_date": "2020-12-15T00:00:00.000+0000",
        "create_log": false
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/card_limit_check"
```

ARGUMENTS

Parameter	Description
`client_id` *UUID, required*	The ID of a Nucleus Client for whom to conduct the analysis.
`currency_code` *string, required*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Only records with this currency code will be considered. See Currencies.
`as_of_date` *datetime*	The date and time as of which to conduct the analysis. Defaults to the current date and time.
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Nucleus as an Audit Log, and the resulting `audit_log_id` will be returned in the Proton response. Audit logs created here can be utilized by the Card Authorization service. Defaults to `false`.

NAV III

Example Response


```
{
    "as_of_date": "2020-12-15T00:00:00.000+0000",
    "currency_code": "USD",
    "summary": {
        "number_of_limits": 6,
        "number_of_limits_breached": 1,
        "number_of_limits_unbreached": 5
    },
    "limit_check": [
        {
            "spending_control_id": "45e1a978-bf92-43b9-a26f-998738622003",
            "control_type": "transaction_limit",
            "control_scope": "purchase",
            "control_values": [],
            "frequency_unit": "daily",
            "frequency": 1,
            "limit_value": 1,
            "limit_used": 1,
            "limit_remaining": 0,
            "limit_breached": true,
            "limit_start_date": "2020-12-15T00:00:00.000000+0000",
            "limit_reset_date": "2020-12-16T00:00:00.000000+0000",
            "days_until_limit_reset": 1,
            "is_primary_transaction_category": [],
            "primary_transaction_category_name": []
        },
        {
            "spending_control_id": "b34c9e4a-66fe-4840-b73a-7b5f3a9dd552",
            "control_type": "spending_limit",
            "control_scope": "purchase",
            "control_values": [],
            "frequency_unit": "annually",
            "frequency": 1,
            "limit_value": 15000.0,
            "limit_used": 2595.4,
            "limit_remaining": 12404.6,
            "limit_breached": false,
            "limit_start_date": "2020-01-01T00:00:00.000000+0000",
            "limit_reset_date": "2021-01-01T00:00:00.000000+0000",
            "days_until_limit_reset": 17,
            "is_primary_transaction_category": [],
            "primary_transaction_category_name": []
```

NAV


```
        },
        {
            "spending_control_id": "591627da-f3d2-4ee0-bb6b-7838dbee3c36",
            "control_type": "spending_limit",
            "control_scope": "atm_withdrawal",
            "control_values": [],
            "frequency_unit": "one_time",
            "frequency": 1,
            "limit_value": 2000.0,
            "limit_used": 0,
            "limit_remaining": 2000.0,
            "limit_breached": false,
            "limit_start_date": null,
            "limit_reset_date": null,
            "days_until_limit_reset": null,
            "is_primary_transaction_category": [],
            "primary_transaction_category_name": []
        },
        {
            "spending_control_id": "60d45207-b17a-4965-bd01-690ffbea0578",
            "control_type": "spending_limit",
            "control_scope": "transaction_category",
            "control_values": [
                "Pets"
            ],
            "frequency_unit": "monthly",
            "frequency": 1,
            "limit_value": 1000.0,
            "limit_used": 253.34,
            "limit_remaining": 746.66,
            "limit_breached": false,
            "limit_start_date": "2020-12-01T00:00:00.000000+0000",
            "limit_reset_date": "2021-01-01T00:00:00.000000+0000",
            "days_until_limit_reset": 17,
            "is_primary_transaction_category": [],
            "primary_transaction_category_name": []
        },
        {
            "spending_control_id": "60d45207-b17a-4965-bd01-690ffbea0578",
            "control_type": "spending_limit",
            "control_scope": "transaction_category",
            "control_values": [
                "Personal Care"
            ],
            "frequency_unit": "monthly",
            "frequency": 1,
```

NAV


```
            "limit_value": 1000.0,
            "limit_used": 47.89,
            "limit_remaining": 952.11,
            "limit_breached": false,
            "limit_start_date": "2020-12-01T00:00:00.000000+0000",
            "limit_reset_date": "2021-01-01T00:00:00.000000+0000",
            "days_until_limit_reset": 17,
            "is_primary_transaction_category": [],
            "primary_transaction_category_name": []
        },
        {
            "spending_control_id": "60d45207-b17a-4965-bd01-690ffbea0578",
            "control_type": "spending_limit",
            "control_scope": "transaction_category",
            "control_values": [
                "Home"
            ],
            "frequency_unit": "monthly",
            "frequency": 1,
            "limit_value": 1000.0,
            "limit_used": 931.62,
            "limit_remaining": 68.38,
            "limit_breached": false,
            "limit_start_date": "2020-12-01T00:00:00.000000+0000",
            "limit_reset_date": "2021-01-01T00:00:00.000000+0000",
            "days_until_limit_reset": 17,
            "is_primary_transaction_category": [],
            "primary_transaction_category_name": []
        }
    ]
}
```


RESPONSE

Field	Description
`as_of_date`	Date and time as of which the analysis is conducted. If `as_of_date` was provided in the request, it is reflected here. If `as_of_date` was not provided, this value is derived dynamically.
`currency_code`	Currency code associated with monetary response values.

NAV

Field	Description
`summary`	Summary of spending and transaction limits for the client.
`number_of_limits`	Total number of limits.
`number_of_limits_breached`	Number of limits that have been breached.
`number_of_limits_unbreached`	Number of limits that have not been breached.
`limit_check`	Detailed analysis of each spending and transaction limit.
`spending_control_id`	ID of the Nucleus Spending Control.
`control_type`	Type of the spending control. Value may be `spending_limit` or `transaction_limit`.
`control_scope`	Scope of the spending control. Value may be `all`, `purchase`, `atm_withdrawal`, `transaction_category`, `transaction_category_id`, `merchant`, `merchant_id`, or `location`.
`control_values`	Value of the spending control. Will be one of the `control_values` from the corresponding spending control.
`frequency_unit`	Frequency unit of the spending control. Value may be `one_time`, `daily`, `weekly`, `monthly`, `quarterly`, or `annually`.
`frequency`	Frequency of the spending control. Indicates the number of `frequency_unit` between each period.
`limit_value`	Value of the spending control limit.
`limit_used`	Value already used towards this spending control limit.
`limit_remaining`	Value that remains to be used towards this spending control limit.
`limit_breached`	Indicates if the limit has been breached. Value of `true` indicates that the limit has been breached, while `false` indicates that the limit has not been breached.

NAV

Field	Description
limit_start_date	Start date of the control period determined by `as_of_date`.
limit_reset_date	End date of the control period determined by `as_of_date`.
days_until_limit_reset	Days remaining in the control period, calculated as the difference between the `as_of_date` and the `limit_reset_date`.
is_primary_transaction_category	If the spending control has `control_scope = transaction_category_id`, indicates whether the `control_value` is a primary `transaction_category_id`. A value of `true` indicates that it is a primary `transaction_category_id`. See Transaction Categories.
primary_transaction_category_name	If the spending control has `control_scope = transaction_category_id` and `control_value` is a primary `transaction_category_id`, then this will be the name of the category. See Transaction Categories.
audit_log_id	The ID of the Audit Log record created in Nucleus. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- Client
- Account
- Portfolio
- Portfolio Asset Size
- Portfolio Transaction
- Transaction Code
- Card
- Spending Control
- Merchants
- Transaction Categories

NAV

PFM & Wealth

Business Financial Management

Cash Analysis

Cash Analysis offers detailed analytics on a business's cash activity across all business accounts. It supports a full view of cash inflows and outflows over a given time period, as well as advanced cash burn metrics (both gross and net). The analysis combines transaction data with balance data to provide insight into where cash is, where it comes from, where it goes, and how each element of cash activity is trending over time.

HTTP REQUEST

```
POST /business/cash_analysis
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "client_id": "dde2a06e-5069-4503-9fa6-19f7f5648a6a",
        "scope": "all",
        "currency_code": "USD",
        "start_date": "2019-08-15",
        "end_date": "2020-08-15",
        "history_frequency_interval": "quarter",
        "show_history": true,
        "show_inflow_details": true,
        "show_outflow_details": true,
        "only_cleansed": true
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/business/cash_analysis
```

ARGUMENTS

NAV

Parameter	Description
`client_id` *UUID*	The ID of a Nucleus Client that represents the business to be analyzed. If not provided, the analysis will include data that is not linked to any client.
`scope` *string*	The scope of data to be considered. Value must be one of the following: `all`, `external`, `internal`. Defaults to `all`, which will consider `internal` data, or Accounts, Portfolios, Portfolio Asset Sizes, and Portfolio Transactions, along with `external` data, or Aggregation Accounts, Aggregation Account Balances, and Aggregation Account Transactions.
`transaction_status_scope` *array[string]*	If populated, only transactions whose `status` matches one of the array values are considered in the analysis. Defaults to `null`, meaning all transactions are considered regardless of the `status`.
`currency_code` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Only records with this currency code will be considered. Defaults to USD if `currency_conversion` is not provided. If `currency_conversion` is provided, all origin currencies will be considered by default. See Currencies.
`currency_conversion` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Records will be converted from their original currency to the one indicated here. Conversions are supported both to and from the following currency codes: USD, GBP, EUR, AUD, CAD, CHF. See Currencies.
`start_date` *date*	The start date of the analysis period. Applies to both balance and transactional data. If there is no underlying data found, `start_date` defaults to one day prior to `end_date`. Else, `start_date` defaults to the earliest date of available data.
`end_date` *date*	The end date of the analysis period. Applies to both balance and transactional data. Defaults to today's date.
`history_frequency_interval` *string*	The frequency of historical cash balance and cash burn. Value can be one of `day`, `week`, `month`, `quarter`, or `year`. For cash balances, when set to a value other than `day`, a historical cash balance entry is returned for the end of each corresponding period. For example, with a value of `month`, the last day of each month in the analysis period will be reflected in the cash balance history. Cash burn history returns

NAV

Parameter	Description
	both a `period_start` and `period_end` date for each interval. Defaults to `month`. Only applicable when `show_history` is `true`.
`show_history` *boolean*	Determines whether to return historical values for cash balance and cash burn analysis. Defaults to `false`.
`show_inflow_details` *boolean*	Determines whether to return details of all cash inflows. Defaults to `false`.
`show_outflow_details` *boolean*	Determines whether to return details of all cash outflows. Defaults to `false`.
`only_cleansed` *boolean*	If `true`, only Portfolio Transactions with the `is_cleansed` parameter set to `true` will be considered. Defaults to `false`.

Example Response


```
{
    "currency_code": "USD",
    "analysis_start": "2019-08-15",
    "analysis_end": "2020-08-15",
    "cash_balance": {
        "current_balance": {
            "value": 41517.85,
            "date": "2020-08-15"
        },
        "mean_balance": 19858.66,
        "median_balance": 20350.0,
        "min_balance": {
            "value": 17950.0,
            "date": "2020-02-04"
        },
        "max_balance": {
            "value": 41517.85,
            "date": "2020-08-15"
        },
        "change": {
            "1_day": {
                "value": 125.0,
                "percentage": 0.003
            },
```

NAV


```
            "7_day": {
                "value": 18917.85,
                "percentage": 0.8371
            },
            "30_day": {
                "value": 19142.85,
                "percentage": 0.8555
            },
            "60_day": {
                "value": 20017.85,
                "percentage": 0.9311
            },
            "90_day": {
                "value": 20292.85,
                "percentage": 0.9561
            },
            "180_day": {
                "value": 21167.85,
                "percentage": 1.0402
            },
            "365_day": {
                "value": 23567.85,
                "percentage": 1.313
            },
            "total": {
                "value": 23567.85,
                "percentage": 1.313
            }
        },
        "history": [
            {
                "date": "2019-12-31",
                "balance": 17950.0,
                "balance_change": {
                    "value": null,
                    "percentage": null
                }
            },
            {
                "date": "2020-08-15",
                "balance": 41517.85,
                "balance_change": {
                    "value": 23567.85,
                    "percentage": 1.313
                }
            }
```

NAV ☰


```
            ]
        },
        "cash_burn": {
            "gross": {
                "current_burn": {
                    "1_day": {
                        "value": 254.24,
                        "start": "2020-08-15",
                        "end": "2020-08-15"
                    },
                    "7_day": {
                        "value": 405.08,
                        "start": "2020-08-09",
                        "end": "2020-08-15"
                    },
                    "30_day": {
                        "value": 1122.28,
                        "start": "2020-07-17",
                        "end": "2020-08-15"
                    },
                    "60_day": {
                        "value": 1724.69,
                        "start": "2020-06-17",
                        "end": "2020-08-15"
                    },
                    "90_day": {
                        "value": 2117.26,
                        "start": "2020-05-18",
                        "end": "2020-08-15"
                    },
                    "180_day": {
                        "value": 2266.09,
                        "start": "2020-02-18",
                        "end": "2020-08-15"
                    },
                    "365_day": {
                        "value": 4766.09,
                        "start": "2019-08-17",
                        "end": "2020-08-15"
                    }
                },
                "mean_burn": {
                    "1_day": 12.99,
                    "7_day": 87.77,
                    "30_day": 377.14,
                    "60_day": 746.36,
```

NAV


```
            "90_day": 1128.29,
            "180_day": 2372.58,
            "365_day": 4596.6
        },
        "median_burn": {
            "1_day": 0.0,
            "7_day": 0.0,
            "30_day": 0.0,
            "60_day": 148.83,
            "90_day": 970.56,
            "180_day": 2500.0,
            "365_day": 4511.85
        },
        "min_burn": {
            "1_day": {
                "value": 0.0,
                "start": "2020-08-14",
                "end": "2020-08-14"
            },
            "7_day": {
                "value": 0.0,
                "start": "2020-08-03",
                "end": "2020-08-09"
            },
            "30_day": {
                "value": 0.0,
                "start": "2020-05-07",
                "end": "2020-06-05"
            },
            "60_day": {
                "value": 0.0,
                "start": "2020-03-03",
                "end": "2020-05-01"
            },
            "90_day": {
                "value": 0.0,
                "start": "2020-02-02",
                "end": "2020-05-01"
            },
            "180_day": {
                "value": 970.56,
                "start": "2020-01-14",
                "end": "2020-07-11"
            },
            "365_day": {
                "value": 4511.85,
```

NAV


```
                    "start": "2019-08-16",
                    "end": "2020-08-14"
                }
            },
            "max_burn": {
                "1_day": {
                    "value": 2500.0,
                    "start": "2020-01-03",
                    "end": "2020-01-03"
                },
                "7_day": {
                    "value": 2500.0,
                    "start": "2020-01-03",
                    "end": "2020-01-09"
                },
                "30_day": {
                    "value": 2500.0,
                    "start": "2020-01-03",
                    "end": "2020-02-01"
                },
                "60_day": {
                    "value": 2500.0,
                    "start": "2020-01-03",
                    "end": "2020-03-02"
                },
                "90_day": {
                    "value": 2500.0,
                    "start": "2020-01-03",
                    "end": "2020-04-01"
                },
                "180_day": {
                    "value": 3470.56,
                    "start": "2020-01-03",
                    "end": "2020-06-30"
                },
                "365_day": {
                    "value": 4766.09,
                    "start": "2019-08-17",
                    "end": "2020-08-15"
                }
            }
        },
        "net": {
            "current_burn": {
                "1_day": {
                    "value": -125.0,
```

NAV III


```
                "start": "2020-08-15",
                "end": "2020-08-15"
            },
            "7_day": {
                "value": -18917.85,
                "start": "2020-08-09",
                "end": "2020-08-15"
            },
            "30_day": {
                "value": -19142.85,
                "start": "2020-07-17",
                "end": "2020-08-15"
            },
            "60_day": {
                "value": -20017.85,
                "start": "2020-06-17",
                "end": "2020-08-15"
            },
            "90_day": {
                "value": -20292.85,
                "start": "2020-05-18",
                "end": "2020-08-15"
            },
            "180_day": {
                "value": -21167.85,
                "start": "2020-02-18",
                "end": "2020-08-15"
            },
            "365_day": {
                "value": -23567.85,
                "start": "2019-08-17",
                "end": "2020-08-15"
            }
        },
        "mean_burn": {
            "1_day": -64.22,
            "7_day": -253.82,
            "30_day": -574.2,
            "60_day": -1003.91,
            "90_day": -1472.58,
            "180_day": -3504.14,
            "365_day": -23501.18
        },
        "median_burn": {
            "1_day": 0.0,
            "7_day": 0.0,
```

NAV


```
            "30_day": 0.0,
            "60_day": -725.0,
            "90_day": -1000.0,
            "180_day": -3275.0,
            "365_day": -23492.85
        },
        "min_burn": {
            "1_day": {
                "value": -17692.85,
                "start": "2020-08-13",
                "end": "2020-08-13"
            },
            "7_day": {
                "value": -18917.85,
                "start": "2020-08-09",
                "end": "2020-08-15"
            },
            "30_day": {
                "value": -19142.85,
                "start": "2020-07-17",
                "end": "2020-08-15"
            },
            "60_day": {
                "value": -20017.85,
                "start": "2020-06-17",
                "end": "2020-08-15"
            },
            "90_day": {
                "value": -20292.85,
                "start": "2020-05-18",
                "end": "2020-08-15"
            },
            "180_day": {
                "value": -21167.85,
                "start": "2020-02-18",
                "end": "2020-08-15"
            },
            "365_day": {
                "value": -23567.85,
                "start": "2019-08-17",
                "end": "2020-08-15"
            }
        },
        "max_burn": {
            "1_day": {
                "value": 850.0,
```

NAV


```
                    "start": "2020-05-10",
                    "end": "2020-05-10"
                },
                "7_day": {
                    "value": 850.0,
                    "start": "2020-05-10",
                    "end": "2020-05-16"
                },
                "30_day": {
                    "value": 850.0,
                    "start": "2020-05-10",
                    "end": "2020-06-08"
                },
                "60_day": {
                    "value": 575.0,
                    "start": "2020-05-06",
                    "end": "2020-07-04"
                },
                "90_day": {
                    "value": 0.0,
                    "start": "2019-11-07",
                    "end": "2020-02-04"
                },
                "180_day": {
                    "value": -2175.0,
                    "start": "2020-02-08",
                    "end": "2020-08-05"
                },
                "365_day": {
                    "value": -23442.85,
                    "start": "2019-08-16",
                    "end": "2020-08-14"
                }
            }
        },
        "history": [
            {
                "period_start": "2019-08-15",
                "period_end": "2019-12-31",
                "gross_burn": 0.0,
                "gross_burn_change": {
                    "value": null,
                    "percentage": null
                },
                "net_burn": 0.0,
                "net_burn_change": {
```

NAV


```
                    "value": null,
                    "percentage": null
                }
            },
            {
                "period_start": "2020-01-01",
                "period_end": "2020-08-15",
                "gross_burn": 4766.09,
                "gross_burn_change": {
                    "value": 4766.09,
                    "percentage": null
                },
                "net_burn": -23567.85,
                "net_burn_change": {
                    "value": -23567.85,
                    "percentage": null
                }
            }
        ]
    },
    "inflow_details": {
        "by_category": [
            {
                "category": "Income",
                "value": 5016.34,
                "subcategories": [
                    {
                        "subcategory": "Deposits",
                        "value": 5000.0
                    },
                    {
                        "subcategory": "Refunds",
                        "value": 16.34
                    }
                ]
            }
        ],
        "by_merchant": [
            {
                "merchant": "Paypal",
                "value": 5000.0
            },
            {
                "merchant": null,
                "value": 16.34
            }
```

NAV


```
            ],
            "outliers": []
        },
        "outflow_details": {
            "by_category": [
                {
                    "category": "Professional Services",
                    "value": 4766.09,
                    "subcategories": [
                        {
                            "subcategory": "Consulting",
                            "value": 2500.0
                        },
                        {
                            "subcategory": "Security",
                            "value": 472.92
                        },
                        {
                            "subcategory": "Printing",
                            "value": 1110.67
                        },
                        {
                            "subcategory": "Marketing",
                            "value": 682.5
                        }
                    ]
                }
            ],
            "by_merchant": [
                {
                    "merchant": "BCG",
                    "value": 2500.0
                },
                {
                    "merchant": "ADT Inc",
                    "value": 472.92
                },
                {
                    "merchant": "FedEx Office",
                    "value": 1110.67
                },
                {
                    "merchant": "ABC Advisors",
                    "value": 682.5
                }
            ],
```

NAV


```
            "outliers": [
                {
                    "id": "48919fbd-2586-4604-9f26-71552dcb82a5",
                    "aggregation_account_id": "d191a596-8031-49a6-b89b-c350e8e3b362",
                    "transaction_date": "2020-01-03T00:00:00.000+0000",
                    "status": null,
                    "bank_credit": {
                        "transaction_type": "Debit",
                        "amount": 2500.0,
                        "merchant": "BCG",
                        "category": "Professional Services",
                        "subcategory": "Consulting",
                        "description": null,
                        "memo": null,
                        "location": {}
                    },
                    "investment": {},
                    "cash": {},
                    "currency_code": "USD",
                    "is_excluded_analysis": false,
                    "is_fee": false,
                    "is_transfer": null,
                    "is_recurring": false,
                    "metadata": {},
                    "secondary_id": null,
                    "create_date": "2020-08-19T19:43:36.618+0000",
                    "update_date": "2020-08-19T19:43:36.618+0000"
                }
            ]
        }
    }
```


RESPONSE

Field	Description
`currency_code`	Currency code associated with monetary response values.
`analysis_start`	Start date of the analysis period. If start_date was provided in the request, it is reflected here. If `start_date` was not provided, this value is derived dynamically.
`analysis_end`	End date of the analysis period. If `end_date` was provided in the request, it is reflected here. If `end_date` was not provided, this value is derived

NAV

Field	Description
	dynamically.
`cash_balance`	Analytics on cash balance activity over time.
`current_balance`	Most recent cash balance, as of the end of the analysis period.
`value`	Cash balance value.
`date`	Effective date for cash balance.
`mean_balance`	Mean daily cash balance over the analysis period.
`median_balance`	Median daily cash balance over the analysis period.
`min_balance`	Minimum daily cash balance over the analysis period.
`value`	Cash balance value.
`date`	Effective date for cash balance.
`max_balance`	Maximum daily cash balance over the analysis period.
`value`	Cash balance value.
`date`	Effective date for cash balance.
`change`	Cash balance change metrics over various periods as of the analysis end date. Only the change periods that are encapsulated in the analysis period are returned.
`1_day`	Change over the past 1 day.
`value`	Absolute balance change.
`percentage`	Percentage balance change.
`7_day`	Change over the past 7 days.
`value`	Absolute balance change.
`percentage`	Percentage balance change.
`30_day`	Change over the past 30 days.

NAV

Field	Description
`value`	Absolute balance change.
`percentage`	Percentage balance change.
`60_day`	Change over the past 60 days.
`value`	Absolute balance change.
`percentage`	Percentage balance change.
`90_day`	Change over the past 90 days.
`value`	Absolute balance change.
`percentage`	Percentage balance change.
`180_day`	Change over the past 180 days.
`value`	Absolute balance change.
`percentage`	Percentage balance change.
`365_day`	Change over the past 365 days.
`value`	Absolute balance change.
`percentage`	Percentage balance change.
`total`	Change over the full analysis period.
`value`	Absolute balance change.
`percentage`	Percentage balance change.
`history`	Historical balance data, with frequency dependent on the `history_frequency_interval` value passed in the request. Each entry has the fields shown below.
`date`	Effective date for cash balance.
`balance`	Cash balance value.
`balance_change`	Change in balance from the prior period.

NAV

Field	Description
`value`	Absolute balance change.
`percentage`	Percentage balance change.
`cash_burn`	Analytics on cash burn over time. Positive burn values indicate decreases in cash, while negative burn values indicate increases in cash.
`gross`	Gross cash burn metrics. Gross reflects cash outflow activity, regardless of inflows. Only the burn periods that are encapsulated in the analysis period are returned.
`current_burn`	Most recent gross burn, as of the end of the analysis period.
`1_day`	Current 1-day cash burn.
`value`	Cash burn value.
`start`	Cash burn start date.
`end`	Cash burn end date.
`7_day`	Current 7-day cash burn.
`value`	Cash burn value.
`start`	Cash burn start date.
`end`	Cash burn end date.
`30_day`	Current 30-day cash burn.
`value`	Cash burn value.
`start`	Cash burn start date.
`end`	Cash burn end date.
`60_day`	Current 60-day cash burn.
`value`	Cash burn value.
`start`	Cash burn start date.

NAV

Field	Description
end	Cash burn end date.
90_day	Current 90-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
180_day	Current 180-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
365_day	Current 365-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
mean_burn	Mean gross cash burn over various periods.
1_day	Mean 1-day cash burn.
7_day	Mean 7-day cash burn.
30_day	Mean 30-day cash burn.
60_day	Mean 60-day cash burn
90_day	Mean 90-day cash burn.
180_day	Mean 180-day cash burn.
365_day	Mean 365-day cash burn.
median_burn	Median gross cash burn over various periods.

NAV

Field	Description
1_day	Median 1-day cash burn.
7_day	Median 7-day cash burn.
30_day	Median 30-day cash burn.
60_day	Median 60-day cash burn.
90_day	Median 90-day cash burn.
180_day	Median 180-day cash burn.
365_day	Median 365-day cash burn.
min_burn	Minimum gross cash burn over various periods.
1_day	Minimum 1-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
7_day	Minimum 7-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
30_day	Minimum 30-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
60_day	Minimum 60-day cash burn.
value	Cash burn value.

NAV

Field	Description
start	Cash burn start date.
end	Cash burn end date.
90_day	Minimum 90-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
180_day	Minimum 180-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
365_day	Minimum 365-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
max_burn	Maximum gross cash burn over various periods.
1_day	Maximum 1-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
7_day	Maximum 7-day cash burn.
value	Cash burn value.
start	Cash burn start date.

NAV

Field	Description
end	Cash burn end date.
30_day	Maximum 30-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
60_day	Maximum 60-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
90_day	Maximum 90-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
180_day	Maximum 180-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
365_day	Maximum 365-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
net	Net cash burn metrics. Net reflects both inflow and outflow cash activity. Only the burn periods that are encapsulated in the analysis period are

NAV

Field	Description
	returned.
`current_burn`	Most recent net burn, as of the end of the analysis period.
`1_day`	Current 1-day cash burn.
`value`	Cash burn value.
`start`	Cash burn start date.
`end`	Cash burn end date.
`7_day`	Current 7-day cash burn.
`value`	Cash burn value.
`start`	Cash burn start date.
`end`	Cash burn end date.
`30_day`	Current 30-day cash burn.
`value`	Cash burn value.
`start`	Cash burn start date.
`end`	Cash burn end date.
`60_day`	Current 60-day cash burn.
`value`	Cash burn value.
`start`	Cash burn start date.
`end`	Cash burn end date.
`90_day`	Current 90-day cash burn.
`value`	Cash burn value.
`start`	Cash burn start date.
`end`	Cash burn end date.

NAV

Field	Description
180_day	Current 180-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
365_day	Current 365-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
mean_burn	Mean net cash burn over various periods.
1_day	Mean 1-day cash burn.
7_day	Mean 7-day cash burn.
30_day	Mean 30-day cash burn.
60_day	Mean 60-day cash burn
90_day	Mean 90-day cash burn.
180_day	Mean 180-day cash burn.
365_day	Mean 365-day cash burn.
median_burn	Median net cash burn over various periods.
1_day	Median 1-day cash burn.
7_day	Median 7-day cash burn.
30_day	Median 30-day cash burn.
60_day	Median 60-day cash burn.
90_day	Median 90-day cash burn.

NAV

Field	Description
180_day	Median 180-day cash burn.
365_day	Median 365-day cash burn.
min_burn	Minimum net cash burn over various periods.
1_day	Minimum 1-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
7_day	Minimum 7-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
30_day	Minimum 30-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
60_day	Minimum 60-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
90_day	Minimum 90-day cash burn.
value	Cash burn value.
start	Cash burn start date.

NAV

Field	Description
end	Cash burn end date.
180_day	Minimum 180-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
365_day	Minimum 365-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
max_burn	Maximum net cash burn over various periods.
1_day	Maximum 1-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
7_day	Maximum 7-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.
30_day	Maximum 30-day cash burn.
value	Cash burn value.
start	Cash burn start date.
end	Cash burn end date.

NAV

Field	Description
`60_day`	Maximum 60-day cash burn.
`value`	Cash burn value.
`start`	Cash burn start date.
`end`	Cash burn end date.
`90_day`	Maximum 90-day cash burn.
`value`	Cash burn value.
`start`	Cash burn start date.
`end`	Cash burn end date.
`180_day`	Maximum 180-day cash burn.
`value`	Cash burn value.
`start`	Cash burn start date.
`end`	Cash burn end date.
`365_day`	Maximum 365-day cash burn.
`value`	Cash burn value.
`start`	Cash burn start date.
`end`	Cash burn end date.
`history`	Historical cash burn data with frequency dependent on the `history_frequency_interval` value passed in the request. Each entry has the fields shown below.
`period_start`	Period start date.
`period_end`	Period end date.
`gross_burn`	Gross cash burn in the period.
`gross_burn_change`	Change in gross cash burn from the prior period.

NAV

Field	Description
value	Absolute gross cash burn change.
percentage	Percentage gross cash burn change.
net_burn	Net cash burn in the period.
net_burn_change	Change in net cash burn from the prior period.
value	Absolute net cash burn change.
percentage	Percentage net cash burn change.
inflow_details	Advanced analytics on cash inflows over the analysis period.
by_category	List of cash inflows separated by categories and their relative subcategories.
category	Category as defined in the Nucleus transactions.
value	Sum of all transactions over the period for the given category.
subcategories	List of cash inflow subcategories as defined in the Nucleus transactions.
subcategory	Subcategory as defined in the Nucleus transactions.
value	Sum of all transactions over the period for the given subcategory.
by_merchant	List of cash inflows separated by merchant.
merchant	Merchant as defined in the Nucleus transactions.
value	Sum of all transactions over the period for the given merchant.
outliers	List of inflow transactions whose amount is beyond 2.5 median absolute deviations from the median of all transaction amounts. Each entry represent a raw transaction record from Nucleus. Please see Nucleus Portfolio Transaction and Aggregation Account Transaction for more details on underlying fields.
outflow_details	Advanced analytics on cash outflows over the analysis period.
by_category	List of cash outflows separated by categories and their relative subcategories.

NAV

Field	Description
category	Category as defined in the Nucleus transactions.
value	Sum of all transactions over the period for the given category.
subcategories	List of cash outflow subcategories as defined in the Nucleus transactions.
subcategory	Subcategory as defined in the Nucleus transactions.
value	Sum of all transactions over the period for the given subcategory.
by_merchant	List of cash outflows separated by merchant.
merchant	Merchant as defined in the Nucleus transactions.
value	Sum of all transactions over the period for the given merchant.
outliers	List of outflow transactions whose amount is beyond 2.5 median absolute deviations from the median of all transaction amounts. Each entry represent a raw transaction record from Nucleus. Please see Nucleus Portfolio Transaction and Aggregation Account Transaction for more details on underlying fields.

NUCLEUS DATA DEPENDENCIES

- Client
- Account
- Portfolio
- Portfolio Asset Size
- Portfolio Transaction
- Aggregation Account
- Aggregation Account Balance
- Aggregation Account Transaction

Customer Analysis

Customer Analysis looks at a business's customers in a revenue context to help frame customer activity in a meaningful way. It offers summary-level revenue metrics, advanced stats on individual payments, a view of the

NAV

firm's top customers, and benchmarked trend indicators. Use this service to keep track of customer revenue over time and to assess the revenue health of a business.

HTTP REQUEST

```
POST /business/customer_analysis
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "client_id": "9e0dd1c0-7669-47ff-9a32-b10434df36dd",
        "currency_code": "USD",
        "start_date": "2020-07-01",
        "end_date": "2020-07-31",
        "benchmark_start_date": "2020-06-01",
        "benchmark_end_date": "2020-06-30",
        "response_limit": 3,
        "only_active_customers": false
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/business/customer_anal
```

ARGUMENTS

Parameter	Description
`client_id` *UUID*	The ID of a Nucleus Client that represents the business to be analyzed. Must have `client_type` of `firm`. If provided, all customers linked to this client will be analyzed. If one of `client_id`, `business_id`, or `customer_ids` is not provided, the analysis will include data that is not linked to any client or business.
`business_id` *UUID*	The ID of a Nucleus Business that represents the business to be analyzed. If provided, all customers linked to this business will be analyzed. If one of `client_id`, `business_id`, or `customer_ids` is not provided, the analysis will include data that is not linked to any client or business.
`customer_ids` *array[UUID]*	The ID(s) of Nucleus Contact(s) to be analyzed. A contact is considered a customer if the `is_customer` parameter is set to `true`. If one of `client_id`, `business_id`, or `customer_ids` is not provided, the analysis will include data that is not linked to any client or business.

NAV

Parameter	Description
`currency_code` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Only records with this currency code will be considered. Defaults to USD if `currency_conversion` is not provided. If `currency_conversion` is provided, all origin currencies will be considered by default. See Currencies.
`currency_conversion` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Records will be converted from their original currency to the one indicated here. Conversions are supported both to and from the following currency codes: USD, GBP, EUR, AUD, CAD, CHF. See Currencies.
`start_date` *date*	Start date of the analysis period. Default to the earliest date of available data.
`end_date` *date*	End date of the analysis period. Defaults to today's date.
`benchmark_start_date` *date*	Start date of the benchmark analysis period. Default values for `benchmark_start_date` and `benchmark_end_date` are based on the difference between `start_date` and `end_date`. For example, if `start_date` and `end_date` are `2016-06-15` and `2016-06-17`, respectively, `benchmark_start_date` and `benchmark_end_date` default to `2016-06-12` and `2016-06-14`, respectively.
`benchmark_end_date` *date*	End date of the benchmark analysis period. Default values for `benchmark_start_date` and `benchmark_end_date` are based on the difference between `start_date` and `end_date`. For example, if `start_date` and `end_date` are `2016-06-15` and `2016-06-17`, respectively, `benchmark_start_date` and `benchmark_end_date` default to `2016-06-12` and `2016-06-14`, respectively.
`response_limit` *integer*	Maximum number of records to be returned in `top_customers`.
`only_active_customers` *boolean*	If `true`, only include Nucleus Customers with `is_active` set to `true` in this analysis. Defaults to `false`, so that both active and inactive customers are included.

Example Response

NAV


```
{
    "currency_code": "USD",
    "base_start_date": "2020-07-01",
    "base_end_date": "2020-07-31",
    "benchmark_start_date": "2020-06-01",
    "benchmark_end_date": "2020-06-30",
    "summary": {
        "number_of_customers": {
            "base": 2,
            "benchmark": 3,
            "change": {
                "value": -1,
                "percentage": -0.3333
            }
        },
        "number_of_payments": {
            "base": 2,
            "benchmark": 4,
            "change": {
                "value": -2,
                "percentage": -0.5
            }
        },
        "total_customer_revenue": {
            "base": 14500.0,
            "benchmark": 24600.0,
            "change": {
                "value": -10100.0,
                "percentage": -0.4106
            }
        }
    },
    "payment_details": {
        "mean_payment": {
            "base": 7250.0,
            "benchmark": 6150.0,
            "change": {
                "value": 1100.0,
                "percentage": 0.1789
            }
        },
        "median_payment": {
            "base": 7250.0,
            "benchmark": 6800.0,
            "change": {
```

NAV


```
                "value": 450.0,
                "percentage": 0.0662
            }
        },
        "min_payment": {
            "base": {
                "company_name": "Initech Printers",
                "date": "2020-07-15",
                "amount": 5750.0
            },
            "benchmark": {
                "company_name": "Cube Financial",
                "date": "2020-06-08",
                "amount": 1500.0
            },
            "change": {
                "value": 4250.0,
                "percentage": 2.8333
            }
        },
        "max_payment": {
            "base": {
                "company_name": "Gradient Entertainment",
                "date": "2020-07-10",
                "amount": 8750.0
            },
            "benchmark": {
                "company_name": "Gradient Entertainment",
                "date": "2020-06-02",
                "amount": 9500.0
            },
            "change": {
                "value": -750.0,
                "percentage": -0.0789
            }
        }
    },
    "top_customers": [
        {
            "company_name": "Gradient Entertainment",
            "number_of_payments": {
                "base": 1,
                "benchmark": 2,
                "change": {
                    "value": -1,
                    "percentage": -0.5
```

NAV


```
                    }
                },
                "total_customer_revenue": {
                    "base": 8750.0,
                    "benchmark": 16825.0,
                    "change": {
                        "value": -8075.0,
                        "percentage": -0.4799
                    }
                },
                "receivable_overdue": 2500.0,
                "receivable_outstanding": 3000.0
            },
            {
                "company_name": "Initech Printers",
                "number_of_payments": {
                    "base": 1,
                    "benchmark": 1,
                    "change": {
                        "value": 0,
                        "percentage": 0.0
                    }
                },
                "total_customer_revenue": {
                    "base": 5750.0,
                    "benchmark": 6275.0,
                    "change": {
                        "value": -525.0,
                        "percentage": -0.0837
                    }
                },
                "receivable_overdue": 7500.0,
                "receivable_outstanding": 12500.0
            }
        ]
    }
```


RESPONSE

Field	Description
`currency_code`	Currency code associated with monetary response values.

NAV

Field	Description
`base_start_date`	Start date of the analysis period. If `start_date` was provided in the request, it is reflected here. If `start_date` was not provided, this value is derived dynamically.
`base_end_date`	End date of the benchmark period. If `end_date` was provided in the request, it is reflected here. If `end_date` was not provided, this value is derived dynamically.
`benchmark_start_date`	Start date of the benchmark period. If `benchmark_start_date` was provided in the request, it is reflected here. If `benchmark_start_date` was not provided, this value is derived dynamically.
`benchmark_end_date`	End date of the benchmark period. If `benchmark_end_date` was provided in the request, it is reflected here. If `benchmark_end_date` was not provided, this value is derived dynamically.
`summary`	High-level analysis of all customer revenue.
`number_of_customers`	Number of customers for which revenue was recorded.
`base`	Number of customers for which revenue was recorded during the base analysis period.
`benchmark`	Number of customers for which revenue was recorded during the benchmark period.
`change`	Difference in number of customers between the base and benchmark periods.
`value`	Absolute difference in number of customers.
`percentage`	Percentage difference in number of customers.
`number_of_payments`	Number of revenue payments made across all customers.
`base`	Number of revenue payments made during the base analysis period.
`benchmark`	Number of revenue payments made during the benchmark analysis period.
`change`	Difference in number of revenue payments between the base and benchmark periods.

NAV

Field	Description
value	Absolute difference in number of revenue payments.
percentage	Percentage difference in number of revenue payments.
total_customer_revenue	Total revenue recorded across all customers.
base	Total revenue recorded during the base analysis period.
benchmark	Total revenue recorded during the benchmark period.
change	Difference in total revenue between the base and benchmark periods.
value	Absolute difference in total revenue.
percentage	Percentage difference in total revenue.
payment_details	Customer revenue payment metrics.
mean_payment	Arithmetic mean payment.
base	Mean payment during the base analysis period.
benchmark	Mean payment during the benchmark analysis period.
change	Difference in mean payment between the base and benchmark periods.
value	Absolute difference in mean payment.
percentage	Percentage difference in mean payment.
median_payment	Median (e.g. 50th percentile) payment.
base	Median payment during the base analysis period.
benchmark	Median payment during the benchmark analysis period.
change	Difference in median payment between the base and benchmark periods.
value	Absolute difference in median payment.
percentage	Percentage difference in median payment.

NAV

Field	Description
`min_payment`	Minimum (e.g. smallest) payment.
`base`	Minimum payment during the base analysis period.
`company_name`	Company name associated with the payment.
`date`	Date of the payment.
`amount`	Amount of the payment.
`benchmark`	Minimum payment during the benchmark period.
`company_name`	Company name associated with the payment.
`date`	Date of the payment.
`amount`	Amount of the payment.
`change`	Difference in minimum payment between the base and benchmark periods.
`value`	Absolute difference in minimum payment.
`percentage`	Percentage difference in minimum payment.
`max_payment`	Maximum (e.g. largest) payment.
`base`	Maximum payment during the base analysis period.
`company_name`	Company name associated with the payment.
`date`	Date of the payment.
`amount`	Amount of the payment.
`benchmark`	Maximum payment during the benchmark period.
`company_name`	Company name associated with the payment.
`date`	Date of the payment.
`amount`	Amount of the payment.

NAV

Field	Description
change	Difference in maximum payment between the base and benchmark periods.
value	Absolute difference in maximum payment.
percentage	Percentage difference in maximum payment.
top_customers	Customers who generated the most revenue over the base analysis period. Each entry represents a single customer and has the fields shown below.
company_name	Company name of the customer.
number_of_payments	Number of revenue payments made by this customer.
base	Number of revenue payments made during the base analysis period.
benchmark	Number of revenue payments made during the benchmark analysis period.
change	Difference in number of revenue payments between the base and benchmark periods.
value	Absolute difference in number of revenue payments.
percentage	Percentage difference in number of revenue payments.
total_customer_revenue	Total revenue recorded for this customer.
base	Total revenue recorded during the base analysis period.
benchmark	Total revenue recorded during the benchmark period.
change	Difference in total revenue between the base and benchmark periods.
value	Absolute difference in total revenue.
percentage	Percentage difference in total revenue.
receivable_overdue	Total receivables for this customer that are overdue.
receivable_outstanding	Total receivables for this customer that are outstanding.

NAV

NUCLEUS DATA DEPENDENCIES

- Client
- Contact
- Customer Revenue

Financial Health Check (Business)

Financial Health Check evaluates the health of a business via a series of key financial ratios. It automatically calculates all available metrics and provides a pass/fail indicator that compares each metric to a rule-of-thumb benchmark. Benchmark values can be configured to further align with a specific business's industry, size, or other unique factors. This tool is designed to provide a high-level snapshot of business financial health.

HTTP REQUEST

```
POST /business/financial_health_check
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "ratio_targets": {
            "profit_margin": 0.1,
            "return_on_assets": 0.05,
            "return_on_equity": 0.15,
            "debt_to_equity": 1.5,
            "debt_ratio": 0.5,
            "asset_turnover": 1.0
        },
        "client_id": "e4a0dc1a-df02-4630-af44-1489f16598a5",
        "currency_code": "USD",
        "accounting_method": "accrual",
        "period_type": "calendar",
        "period_length": "quarterly"
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/business/financial_hea
```

ARGUMENTS

NAV

Parameter	Description
`ratio_targets` *map*	Benchmark values for each of the calculated metrics. Used to perform comparative analysis.
`profit_margin` *float*	Benchmark value for Profit Margin. Defaults to 0.1.
`return_on_assets` *float*	Benchmark value for Return on Assets. Defaults to 0.05.
`return_on_equity` *float*	Benchmark value for Return on Equity. Defaults to 0.15.
`debt_to_equity` *float*	Benchmark value for Debt-to-Equity ratio. Defaults to 1.5.
`debt_ratio` *float*	Benchmark value for Debt Ratio. Defaults to 0.5.
`asset_turnover` *float*	Benchmark value for Asset Turnover. Defaults to 1.0.
`net_income` *float*	Raw input for net income. If provided, takes precedent over data sourced from Nucleus.
`total_revenue` *float*	Raw input for total revenue. If provided, takes precedent over data sourced from Nucleus.
`total_assets` *float*	Raw input for total assets. If provided, takes precedent over data sourced from Nucleus.
`total_liabilities` *float*	Raw input for total liabilities. If provided, takes precedent over data sourced from Nucleus.
`total_equity` *float*	Raw input for total shareholder's equity. If provided, takes precedent over data sourced from Nucleus.
`client_id` *UUID*	The ID of a Nucleus Client that represents the business to be analyzed. Must have `client_type` of `firm`. If neither a `client_id`, `business_id`, or raw values are provided for each input, the analysis will include data that is not linked to any client or business.

NAV

Parameter	Description
`business_id` *UUID*	The ID of a Nucleus Business that represents the business to be analyzed. If neither a `client_id`, `business_id`, or raw values are provided for each input, the analysis will include data that is not linked to any client or business.
`currency_code` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Only records with this currency code will be considered. Defaults to USD if `currency_conversion` is not provided. If `currency_conversion` is provided, all origin currencies will be considered by default. See Currencies.
`currency_conversion` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Records will be converted from their original currency to the one indicated here. Conversions are supported both to and from the following currency codes: USD, GBP, EUR, AUD, CAD, CHF. See Currencies.
`accounting_method` *string*	The financial statement accounting method to consider. Value may be `accrual` or `cash`. If not provided, Nucleus records with any period type will be considered.
`period_type` *string*	The income statement period type to consider. Value may be `calendar` or `fiscal`. If not provided, Nucleus records with any period type will be considered.
`period_length` *string*	The income statement period length to consider. Value may be `annual`, `quarterly`, `monthly`, `ytd`, `qtd`, or `mtd`. Defaults to `quarterly`.
`period_year` *integer*	The income statement period length to consider. If not provided, defaults to the most recent available year found in an applicable financial statement.
`period_quarter` *integer*	The income statement period quarter to consider. If not provided, defaults to the most recent available quarter found in an applicable financial statement.
`period_month` *integer*	The income statement period month to consider. If not provided, defaults to the most recent available quarter found in an applicable financial statement.

Example Response

```
{
    "currency_code": "USD",
    "profit_margin": {
```

NAV


```
            "value": 0.4792,
            "target": 0.1,
            "pass": true
        },
        "return_on_assets": {
            "value": 0.03,
            "target": 0.05,
            "pass": false
        },
        "return_on_equity": {
            "value": 0.2812,
            "target": 0.15,
            "pass": true
        },
        "debt_to_equity": {
            "value": 0.32,
            "target": 1.5,
            "pass": true
        },
        "debt_ratio": {
            "value": 0.1067,
            "target": 0.05,
            "pass": true
        },
        "asset_turnover": {
            "value": 0.0626,
            "target": 1.0,
            "pass": false
        },
        "net_income": 11248.23,
        "total_revenue": 23474.12,
        "total_assets": 375000.0,
        "total_liabilities": 40000.0,
        "total_equity": 125000.0
    }
```


RESPONSE

Field	Description
`currency_code`	Currency code associated with monetary response values.
`profit_margin`	Results for Profit Margin.
`value`	The calculated value.

NAV

Field	Description
`target`	The target value, as determined by `ratio_targets`.
`pass`	If `true`, `value` is sufficient as compared to its target.
`return_on_assets`	Results for Return on Assets.
`value`	The calculated value.
`target`	The target value, as determined by `ratio_targets`.
`pass`	If `true`, `value` is sufficient as compared to its target.
`return_on_equity`	Results for Return on Equity.
`value`	The calculated value.
`target`	The target value, as determined by `ratio_targets`.
`pass`	If `true`, `value` is sufficient as compared to its target.
`debt_to_equity`	Results for Debt-to-Equity.
`value`	The calculated value.
`target`	The target value, as determined by `ratio_targets`.
`pass`	If `true`, `value` is sufficient as compared to its target.
`debt_ratio`	Results for Debt Ratio.
`value`	The calculated value.
`target`	The target value, as determined by `ratio_targets`.
`pass`	If `true`, `value` is sufficient as compared to its target.
`asset_turnover`	Results for Asset Turnover.
`value`	The calculated value.
`target`	The target value, as determined by `ratio_targets`.
`pass`	If `true`, `value` is sufficient as compared to its target.

NAV

Field	Description
`net_income`	The underlying net income used for the analysis. If `net_income` is not provided in the request, this value is derived dynamically.
`total_revenue`	The underlying total revenue used for the analysis. If `total_revenue` is not provided in the request, this value is derived dynamically.
`total_assets`	The underlying total assets used for the analysis. If `total_assets` is not provided in the request, this value is derived dynamically.
`total_liabilities`	The underlying total liabilities used for the analysis. If `total_liabilities` is not provided in the request, this value is derived dynamically.
`total_equity`	The underlying total equity used for the analysis. If `total_equity` is not provided in the request, this value is derived dynamically.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Financial Statement`

Financial Statement Analysis

Financial Statement Analysis provides a structured view of a business's accounting data over time. The analysis includes metrics from both the Income Statement (e.g. Net Income) and Balance Sheet (e.g. Total Assets). This tool is helpful for assessing trends in business activity via historical and statistical analysis, and also acts as an interface to view a high-level snapshot of business performance.

HTTP REQUEST

`POST /business/financial_statement_analysis`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
```

NAV

```
        "client_id": "e4a0dc1a-df02-4630-af44-1489f16598a5",
        "currency_code": "USD",
        "stat_names": [
            "total_assets",
            "total_liabilities",
            "total_equity",
            "total_revenue",
            "net_income"
    ],
        "accounting_method": "accrual",
        "period_type": "calendar",
        "period_length": "quarterly",
        "period_year": 2020,
        "period_quarter": 1,
        "start_date": "2020-05-31",
        "end_date": "2020-08-31",
        "show_history": true,
        "history_frequency_interval": "month"
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/business/financial_sta
```

ARGUMENTS

Parameter	Description
`client_id` *UUID*	The ID of a Nucleus Client that represents the business to be analyzed. Must have `client_type` of `firm`. If one of `client_id` or `business_id` is not provided, the analysis will include data that is not linked to any client or business.
`business_id` *UUID*	The ID of a Nucleus Business that represents the business to be analyzed. If one of `client_id` or `business_id` is not provided, the analysis will include data that is not linked to any client or business.
`currency_code` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Only records with this currency code will be considered. Defaults to USD if `currency_conversion` is not provided. If `currency_conversion` is provided, all origin currencies will be considered by default. See Currencies.
`currency_conversion` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Records will be converted from their original currency to the one indicated here. Conversions are supported both to and from

NAV

Parameter	Description
	the following currency codes: USD, GBP, EUR, AUD, CAD, CHF. See Currencies.
`start_date` *date*	Start date for balance sheet analysis. Defaults to the earliest financial statement date available.
`end_date` *date*	End date for balance sheet analysis. Defaults to today's date.
`statement_type` *string*	The type of financial statement to include in the analysis. Value may be `balance_sheet`, `income_statement`, or `all`. Default to `all`. Please note that `statement_type` and `stat_names` both contribute to the scope of the analysis.
`stat_names` *array[string]*	Names of specific financial stats to include in the analysis. Values may be from the income statement (`total_revenue`, `cogs`, `gross_profit`, `operating_income`, `total_expenses`, `ebitda`, `net_income`), or the balance sheet (`cash`, `total_assets`, `total_liabilities`, `total_equity`, `accounts_receivable`, `accounts_payable`). If not provided, all stats within the scope of `statement_type` are considered.
`accounting_method` *string, required*	The accounting method for the analysis. Value may be `accrual` or `cash`. Only financial statements with this accounting method are considered.
`period_type` *string, conditionally required*	The period type to consider for income statement analysis if `statement_type = income_statement` or any value in `stat_names` falls under income statement. Value may be `calendar` or `fiscal`.
`period_length` *string*	The period length to consider for income statement analysis if `statement_type = income_statement` or any value in `stat_names` falls under income statement. Value may be `annual`, `quarterly`, `monthly`, `ytd`, `qtd`, or `mtd`. Defaults to `quarterly`.
`period_year` *integer*	Year to consider for income statement analysis if `statement_type = income_statement` or any value in `stat_names` falls under income statement. If not provided, defaults to the most recent available year found in an applicable financial statement.
`period_quarter` *integer*	Quarter to consider for income statement analysis if `statement_type = income_statement` or any value in `stat_names` falls under

NAV

Parameter	Description
	statement. If not provided, defaults to the most recent available quarter found in an applicable financial statement.
`period_month` *integer*	Month to consider for income statement analysis if `statement_type = income_statement` or any value in `stat_names` falls under income statement. If not provided, defaults to the most recent available month found in an applicable financial statement.
`history_frequency_interval` *string*	The frequency of historical stat values for balance sheet analysis. Value can be one of `day`, `week`, `month`, `quarter`, or `year`. When set to a value other than `day`, a historical entry is returned for the end of each corresponding period. For example, with a value of `month`, the last day of each month in the analysis period will be reflected in the stat history. Defaults to `month`. Only applicable when `show_history` is `true`.
`show_history` *boolean*	Determines whether to return historical stat values for balance sheet analysis. Defaults to `false`.

Example Response


```
{
    "currency_code": "USD",
    "accounting_method": "accrual",
    "balance_sheet": {
        "analysis_start": "2020-05-31",
        "analysis_end": "2020-08-31",
        "stats": [
            {
                "name": "total_assets",
                "current_value": 375000.0,
                "mean_value": 309408.6,
                "median_value": 300000.0,
                "min_value": {
                    "value": 275000.0,
                    "date": "2020-06-29"
                },
                "max_value": {
                    "value": 375000.0,
                    "date": "2020-08-31"
                },
                "change": {
```

NAV


```
                "1_day": {
                    "value": 25000.0,
                    "percentage": 0.0714
                },
                "7_day": {
                    "value": 25000.0,
                    "percentage": 0.0714
                },
                "30_day": {
                    "value": 25000.0,
                    "percentage": 0.0714
                },
                "60_day": {
                    "value": 75000.0,
                    "percentage": 0.25
                },
                "90_day": {
                    "value": 100000.0,
                    "percentage": 0.3636
                },
                "total": {
                    "value": 100000.0,
                    "percentage": 0.3636
                }
            }
        },
        {
            "name": "total_equity",
            "current_value": 125000.0,
            "mean_value": 80645.16,
            "median_value": 75000.0,
            "min_value": {
                "value": 65000.0,
                "date": "2020-06-29"
            },
            "max_value": {
                "value": 125000.0,
                "date": "2020-08-31"
            },
            "change": {
                "1_day": {
                    "value": 25000.0,
                    "percentage": 0.25
                },
                "7_day": {
                    "value": 25000.0,
```

NAV III


```
                "percentage": 0.25
            },
            "30_day": {
                "value": 25000.0,
                "percentage": 0.25
            },
            "60_day": {
                "value": 50000.0,
                "percentage": 0.6667
            },
            "90_day": {
                "value": 60000.0,
                "percentage": 0.9231
            },
            "total": {
                "value": 60000.0,
                "percentage": 0.9231
            }
        }
    },
    {
        "name": "total_liabilities",
        "current_value": 40000.0,
        "mean_value": 51451.61,
        "median_value": 50000.0,
        "min_value": {
            "value": 40000.0,
            "date": "2020-08-31"
        },
        "max_value": {
            "value": 60000.0,
            "date": "2020-06-29"
        },
        "change": {
            "1_day": {
                "value": -5000.0,
                "percentage": -0.1111
            },
            "7_day": {
                "value": -5000.0,
                "percentage": -0.1111
            },
            "30_day": {
                "value": -5000.0,
                "percentage": -0.1111
            },
```

NAV


```
                "60_day": {
                    "value": -10000.0,
                    "percentage": -0.2
                },
                "90_day": {
                    "value": -20000.0,
                    "percentage": -0.3333
                },
                "total": {
                    "value": -20000.0,
                    "percentage": -0.3333
                }
            }
        }
    ],
    "history": [
        {
            "date": "2020-05-31",
            "values": [
                {
                    "name": "total_assets",
                    "value": 275000.0
                },
                {
                    "name": "total_liabilities",
                    "value": 60000.0
                },
                {
                    "name": "total_equity",
                    "value": 65000.0
                }
            ]
        },
        {
            "date": "2020-06-30",
            "values": [
                {
                    "name": "total_assets",
                    "value": 300000.0
                },
                {
                    "name": "total_liabilities",
                    "value": 50000.0
                },
                {
                    "name": "total_equity",
```

NAV


```
                        "value": 75000.0
                    }
                ]
            },
            {
                "date": "2020-07-31",
                "values": [
                    {
                        "name": "total_assets",
                        "value": 350000.0
                    },
                    {
                        "name": "total_liabilities",
                        "value": 45000.0
                    },
                    {
                        "name": "total_equity",
                        "value": 100000.0
                    }
                ]
            },
            {
                "date": "2020-08-31",
                "values": [
                    {
                        "name": "total_assets",
                        "value": 375000.0
                    },
                    {
                        "name": "total_liabilities",
                        "value": 40000.0
                    },
                    {
                        "name": "total_equity",
                        "value": 125000.0
                    }
                ]
            }
        ]
    },
    "income_statement": {
        "period_length": "quarterly",
        "period_type": "calendar",
        "statement_details": {
            "statement_date": "2020-03-31",
            "period_year": 2020,
```

NAV


```
                "period_quarter": 1,
                "period_month": null
            },
            "previous_statement_details": {
                "statement_date": "2019-12-31",
                "period_year": 2019,
                "period_quarter": 4,
                "period_month": null
            },
            "stats": [
                {
                    "name": "total_revenue",
                    "value": 23474.12,
                    "previous_value": 19264.32,
                    "change": {
                        "value": 4209.8,
                        "percentage": 0.2185
                    }
                },
                {
                    "name": "net_income",
                    "value": 11248.23,
                    "previous_value": 9784.75,
                    "change": {
                        "value": 1463.48,
                        "percentage": 0.1496
                    }
                }
            ]
        }
    }
```


RESPONSE

Field	Description
`currency_code`	Currency code associated with monetary response values.
`accounting_method`	Accounting method associated with response values.
`balance_sheet`	Analytics on balance sheet stats.
`analysis_start`	Start date of the analysis period. If `start_date` was provided in the request, it is reflected here. If `start_date` was not provided,

NAV

Field	Description
	this value is derived dynamically.
`analysis_end`	End date of the analysis. If `end_date` was provided in the request, it is reflected here. If `end_date` was not provided, this value is derived dynamically.
`stats`	Metrics on each stat included in the analysis. Each entry has the fields shown below.
`name`	Name of the stat.
`current_value`	Most recent stat value, as of the end of the analysis period.
`mean_value`	Mean stat value over the analysis period.
`median_value`	Median stat value over the analysis period.
`min_value`	Minimum stat value over the analysis period.
`value`	Stat value.
`date`	Effective date for stat value.
`max_value`	Maximum stat value over the analysis period.
`value`	Stat value.
`date`	Effective date for stat value.
`change`	Stat change metrics over various periods, as of the `analysis_end_date`. Only the change periods that are encapsulated in the analysis period are returned.
`1_day`	Change over the past 1 day.
`value`	Absolute change.
`percentage`	Percentage change.
`7_day`	Change over the past 7 days.
`value`	Absolute change.

NAV

Field	Description
`percentage`	Percentage change.
`30_day`	Change over the past 30 days.
`value`	Absolute change.
`percentage`	Percentage change.
`60_day`	Change over the past 60 days.
`value`	Absolute change.
`percentage`	Percentage change.
`90_day`	Change over the past 90 days.
`value`	Absolute change.
`percentage`	Percentage change.
`180_day`	Change over the past 180 days.
`value`	Absolute change.
`percentage`	Percentage change.
`365_day`	Change over the past 365 days.
`value`	Absolute change.
`percentage`	Percentage change.
`total`	Change over the full analysis period.
`value`	Absolute change.
`percentage`	Percentage change.
`history`	Historical stat data, with frequency dependent on the `history_frequency_interval` value passed in the request. Each entry has the fields shown below.
`date`	Effective date for stat values.

NAV

Field	Description
values	Historical values for each stat included in the analysis. Each entry has the fields shown below.
name	Name of the stat.
value	Historical stat value.
income_statement	Analytics on income statement stats.
period_length	Periodicity of the analysis.
period_type	Period type of the analysis.
statement_details	Details regarding the primary statement in the analysis.
statement_date	Recorded statement date.
period_year	Year associated with the statement.
period_quarter	Quarter associated with the statement, if `period_length` is `quarterly` or `qtd`.
period_month	Month associated with the statement, if `period_length` is `monthly` or `mtd`.
previous_statement_details	Details regarding the previous statement in the analysis.
statement_date	Recorded statement date.
period_year	Year associated with the statement.
period_quarter	Quarter associated with the statement, if `period_length` is `quarterly` or `qtd`.
period_month	Month associated with the statement, if `period_length` is `monthly` or `mtd`.
stats	Metrics on each stat included in the analysis. Each entry has the fields shown below.
name	Name of the stat.
value	Stat value from the primary statement in the analysis.

NAV

Field	Description
`previous_value`	Stat value from the previous statement in the analysis.
`change`	Stat change from the previous statement to the primary statement.
`value`	Absolute change.
`name`	Percentage change.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Financial Statement`

Invoice Analysis

Invoice Analysis produces advanced analytics on invoice activity for a business. This includes a high-level summary of both receivable and payable invoices issued and paid, as well as information on outstanding invoices, current invoices, overdue invoices, and invoices coming due in configurable future time windows. It also highlights invoiced customers and suppliers, as well as individual invoices that may warrant a business's attention. Use this service to gain valuable insights into both current and historical invoice activity.

HTTP REQUEST

`POST /business/invoice_analysis`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "client_id": "9e0dd1c0-7669-47ff-9a32-b10434df36dd",
        "currency_code": "USD",
        "start_date": "2020-01-01",
        "end_date": "2020-08-31",
        "history_frequency_interval": "month",
        "response_limit": 1,
```

NAV

```
  "show_history": true,
  "show_outstanding_invoices": true
}' "https://[sandbox][api].hydrogenplatform.com/proton/v1/business/invoice_analy
```

ARGUMENTS

Parameter	Description
`client_id` *UUID*	The ID of a Nucleus Client that represents the business to be analyzed. Must have `client_type` of `firm`. If one of `client_id`, `business_id`, `customer_ids`, or `invoice_ids` is not provided, the analysis will include data that is not linked to any client or business.
`business_id` *UUID*	The ID of a Nucleus Business that represents the business to be analyzed. If one of `client_id`, `business_id`, `customer_ids`, or `invoice_ids` is not provided, the analysis will include data that is not linked to any client or business.
`customer_ids` *array[UUID]*	The ID(s) of Nucleus Contact(s) to be analyzed. A contact either is considered a customer if the `is_customer` parameter is set to `true`, or is considered a supplier if the `is_supplier` parameter is set to `true`. If one of `client_id`, `business_id`, `customer_ids`, or `invoice_ids` is not provided, the analysis will include data that is not linked to any client or business.
`invoice_ids` *array[UUID]*	The ID(s) of Nucleus Invoice(s) to be analyzed. If one of `client_id`, `business_id`, `customer_ids`, or `invoice_ids` is not provided, the analysis will include data that is not linked to any client or business.
`currency_code` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Only records with this currency code will be considered. Defaults to USD if `currency_conversion` is not provided. If `currency_conversion` is provided, all origin currencies will be considered by default. See Currencies.
`currency_conversion` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Records will be converted from their original currency to the one indicated here. Conversions are supported both to and from the following currency codes: USD, GBP, EUR, AUD, CAD, CHF. See Currencies.

NAV

Parameter	Description
`start_date` *date*	Start date for invoice analysis. Defaults to the earliest date of available data. If there is no underlying data found, `start_date` defaults to one day prior to `end_date`.
`end_date` *date*	End date for invoice analysis. Defaults to today's date.
`future_due_periods` *array[map]*	Periods in days relative to `end_date` that apply to `future_due_periods` in the response. Defaults to show 30 day increments. Each entry has the fields shown below. Duplicate entries may not be passed.
`start` *integer*	Start day of the period, such as 1. Must be greater than or equal to 0.
`end` *integer*	End day of the period, such as 30. Must be greater than or equal to `start`. Use `null` to indicate all integers greater than `start`.
`overdue_periods` *array[map]*	Periods in days relative to `end_date` that apply to `overdue_periods` in the response. Defaults to show 30 day increments. Each entry has the fields shown below. Duplicate entries may not be passed.
`start` *integer*	Start day of the period, such as 1. Must be greater than or equal to 0.
`end` *integer*	End day of the period, such as 30. Must be greater than or equal to `start`. Use `null` to indicate all integers greater than `start`.
`response_limit` *integer*	The maximum number of records to be returned in `top_overdue_invoices`, `top_overdue_customers`, `top_paid_invoices`, and `top_paid_customers`.
`history_frequency_interval` *string*	The frequency of historical invoice data. Defaults to `month`. Only applicable when `show_history` is `true`.
`show_history` *boolean*	If `true`, returns historical invoice data. Defaults to `false`.
`show_outstanding_invoices` *boolean*	If `true`, returns details on all outstanding invoices. Defaults to `false`.

NAV III

Example Response


```
{
    "currency_code": "USD",
    "analysis_start": "2020-01-01",
    "analysis_end": "2020-08-31",
    "receivable": {
        "total_number_of_invoices": 9,
        "total_amount_invoiced": 74703.07,
        "total_amount_paid": 27089.85,
        "total_amount_unpaid": 47613.22,
        "paid_summary": {
            "number_of_invoices": 2,
            "amount_paid": 12500.0
        },
        "outstanding_summary": {
            "number_of_invoices": 7,
            "amount_paid": 14589.85,
            "amount_unpaid": 47613.22
        },
        "current_summary": {
            "number_of_invoices": 2,
            "amount_paid": 683.29,
            "amount_unpaid": 12077.2,
            "percent_of_outstanding": 0.2537
        },
        "overdue_summary": {
            "number_of_invoices": 5,
            "amount_paid": 13906.56,
            "amount_unpaid": 35536.02,
            "percent_of_outstanding": 0.7463
        },
        "future_due_periods": [
            {
                "start": 0,
                "end": 30,
                "number_of_invoices": 1,
                "amount_paid": 683.29,
                "amount_unpaid": 3644.26,
                "percent_of_current": 0.3017
            },
            {
                "start": 31,
                "end": 60,
```

NAV III


```
                "number_of_invoices": 1,
                "amount_paid": 0,
                "amount_unpaid": 8432.94,
                "percent_of_current": 0.6983
            },
            {
                "start": 61,
                "end": 90,
                "number_of_invoices": 0,
                "amount_paid": 0,
                "amount_unpaid": 0,
                "percent_of_current": 0.0
            },
            {
                "start": 91,
                "end": null,
                "number_of_invoices": 0,
                "amount_paid": 0,
                "amount_unpaid": 0,
                "percent_of_current": 0.0
            }
        ],
        "overdue_periods": [
            {
                "start": 1,
                "end": 30,
                "number_of_invoices": 2,
                "amount_paid": 2750.0,
                "amount_unpaid": 10571.24,
                "percent_of_overdue": 0.2975
            },
            {
                "start": 31,
                "end": 60,
                "number_of_invoices": 1,
                "amount_paid": 1500.0,
                "amount_unpaid": 6000.0,
                "percent_of_overdue": 0.1688
            },
            {
                "start": 61,
                "end": 90,
                "number_of_invoices": 1,
                "amount_paid": 3000.0,
                "amount_unpaid": 12000.0,
                "percent_of_overdue": 0.3377
```

NAV


```
            },
            {
                "start": 91,
                "end": null,
                "number_of_invoices": 1,
                "amount_paid": 6656.56,
                "amount_unpaid": 6964.78,
                "percent_of_overdue": 0.196
            }
        ],
        "top_overdue_invoices": [
            {
                "invoice_id": "f83fc955-9509-46e0-9d8b-514b578e8351",
                "invoice_number": "3647031",
                "invoice_date": "2020-05-12",
                "invoice_due_date": "2020-06-12",
                "days_overdue": 80,
                "amount_invoiced": 15000.0,
                "amount_paid": 3000.0,
                "amount_unpaid": 12000.0,
                "percent_of_overdue": 0.3377
            }
        ],
        "top_overdue_customers": [
            {
                "contact_id": "c387c95c-657f-4334-a947-9590242ab742",
                "company_name": "Cube Financial",
                "number_of_invoices": 2,
                "amount_paid": 9656.56,
                "amount_unpaid": 18964.78,
                "overdue_periods": [
                    {
                        "start": 1,
                        "end": 30,
                        "number_of_invoices": 0,
                        "amount_paid": 0,
                        "amount_unpaid": 0,
                        "percent_of_overdue": 0.0
                    },
                    {
                        "start": 31,
                        "end": 60,
                        "number_of_invoices": 0,
                        "amount_paid": 0,
                        "amount_unpaid": 0,
                        "percent_of_overdue": 0.0
```

NAV


```
                    },
                    {
                        "start": 61,
                        "end": 90,
                        "number_of_invoices": 1,
                        "amount_paid": 3000.0,
                        "amount_unpaid": 12000.0,
                        "percent_of_overdue": 0.6328
                    },
                    {
                        "start": 91,
                        "end": null,
                        "number_of_invoices": 1,
                        "amount_paid": 6656.56,
                        "amount_unpaid": 6964.78,
                        "percent_of_overdue": 0.3672
                    }
                ]
            }
        ],
        "top_paid_invoices": [
            {
                "invoice_id": "9dd7f0c2-5728-4a4f-9447-1c5c8d53c194",
                "invoice_number": "3244929",
                "invoice_date": "2020-02-01",
                "amount_paid": 10000.0,
                "percent_of_paid": 0.8
            }
        ],
        "top_paid_customers": [
            {
                "contact_id": "b4407cfa-fbae-4cd9-9ad5-71c0bcdad286",
                "company_name": "Gradient Entertainment",
                "number_of_invoices": 3,
                "amount_paid": 14250.0
            }
        ],
        "history": [
            {
                "period_start": "2020-01-01",
                "period_end": "2020-01-31",
                "amount_invoiced": 0.0,
                "amount_paid": 0.0
            },
            {
                "period_start": "2020-02-01",
```

NAV


```
                "period_end": "2020-02-29",
                "amount_invoiced": 10000.0,
                "amount_paid": 0.0
            },
            {
                "period_start": "2020-03-01",
                "period_end": "2020-03-31",
                "amount_invoiced": 0.0,
                "amount_paid": 10000.0
            },
            {
                "period_start": "2020-04-01",
                "period_end": "2020-04-30",
                "amount_invoiced": 13621.34,
                "amount_paid": 0.0
            },
            {
                "period_start": "2020-05-01",
                "period_end": "2020-05-31",
                "amount_invoiced": 15000.0,
                "amount_paid": 1452.57
            },
            {
                "period_start": "2020-06-01",
                "period_end": "2020-06-30",
                "amount_invoiced": 7500.0,
                "amount_paid": 4783.92
            },
            {
                "period_start": "2020-07-01",
                "period_end": "2020-07-31",
                "amount_invoiced": 15821.24,
                "amount_paid": 4346.21
            },
            {
                "period_start": "2020-08-01",
                "period_end": "2020-08-31",
                "amount_invoiced": 12760.49,
                "amount_paid": 4007.15
            }
        ],
        "outstanding_invoices": [
            {
                "invoice_id": "f83fc955-9509-46e0-9d8b-514b578e8351",
                "invoice_number": "3647031",
                "invoice_date": "2020-05-12",
```

NAV


```
            "invoice_due_date": "2020-06-12",
            "days_until_due": null,
            "days_overdue": 80,
            "is_overdue": true,
            "number_of_payments": 1,
            "latest_payment_date": "2020-06-25",
            "amount_invoiced": 15000.0,
            "amount_paid": 3000.0,
            "amount_unpaid": 12000.0
        },
        {
            "invoice_id": "4af61b65-8499-49f8-8853-602e1b6f105d",
            "invoice_number": "5240307",
            "invoice_date": "2020-08-26",
            "invoice_due_date": "2020-10-26",
            "days_until_due": 56,
            "days_overdue": null,
            "is_overdue": false,
            "number_of_payments": 0,
            "latest_payment_date": null,
            "amount_invoiced": 8432.94,
            "amount_paid": 0,
            "amount_unpaid": 8432.94
        },
        {
            "invoice_id": "bb26fdfb-0b3b-4ec1-b127-054b703bfa39",
            "invoice_number": "4981774",
            "invoice_date": "2020-07-26",
            "invoice_due_date": "2020-08-27",
            "days_until_due": null,
            "days_overdue": 4,
            "is_overdue": true,
            "number_of_payments": 0,
            "latest_payment_date": null,
            "amount_invoiced": 8321.24,
            "amount_paid": 0,
            "amount_unpaid": 8321.24
        },
        {
            "invoice_id": "00bef7eb-8127-4bff-ba60-eece19cf3bd7",
            "invoice_number": "8600501",
            "invoice_date": "2020-04-18",
            "invoice_due_date": "2020-05-18",
            "days_until_due": null,
            "days_overdue": 105,
            "is_overdue": true,
```

NAV III


```
            "number_of_payments": 4,
            "latest_payment_date": "2020-05-28",
            "amount_invoiced": 13621.34,
            "amount_paid": 6656.56,
            "amount_unpaid": 6964.78
        },
        {
            "invoice_id": "54cf6152-6de0-4e18-9e67-a1da59831a6f",
            "invoice_number": "5861978",
            "invoice_date": "2020-06-22",
            "invoice_due_date": "2020-07-22",
            "days_until_due": null,
            "days_overdue": 40,
            "is_overdue": true,
            "number_of_payments": 1,
            "latest_payment_date": "2020-07-31",
            "amount_invoiced": 7500.0,
            "amount_paid": 1500.0,
            "amount_unpaid": 6000.0
        },
        {
            "invoice_id": "bd6c3f2d-72cd-43ae-b3a7-c97c474fc39b",
            "invoice_number": "8188898",
            "invoice_date": "2020-08-01",
            "invoice_due_date": "2020-09-30",
            "days_until_due": 30,
            "days_overdue": null,
            "is_overdue": false,
            "number_of_payments": 1,
            "latest_payment_date": "2020-08-19",
            "amount_invoiced": 4327.55,
            "amount_paid": 683.29,
            "amount_unpaid": 3644.26
        },
        {
            "invoice_id": "fd895006-a54a-406b-9acf-21bc91e443e2",
            "invoice_number": "9429762",
            "invoice_date": "2020-07-16",
            "invoice_due_date": "2020-08-16",
            "days_until_due": null,
            "days_overdue": 15,
            "is_overdue": true,
            "number_of_payments": 1,
            "latest_payment_date": "2020-08-19",
            "amount_invoiced": 5000.0,
            "amount_paid": 2750.0,
```

NAV III


```
                "amount_unpaid": 2250.0
            }
        ]
    },
    "payable": {
        "total_number_of_invoices": 3,
        "total_amount_invoiced": 48331.63,
        "total_amount_paid": 16425.84,
        "total_amount_unpaid": 31905.79,
        "paid_summary": {
            "number_of_invoices": 1,
            "amount_paid": 5000.0
        },
        "outstanding_summary": {
            "number_of_invoices": 2,
            "amount_paid": 11425.84,
            "amount_unpaid": 31905.79
        },
        "current_summary": {
            "number_of_invoices": 1,
            "amount_paid": 6734.56,
            "amount_unpaid": 18703.86,
            "percent_of_outstanding": 0.5862
        },
        "overdue_summary": {
            "number_of_invoices": 1,
            "amount_paid": 4691.28,
            "amount_unpaid": 13201.93,
            "percent_of_outstanding": 0.4138
        },
        "future_due_periods": [
            {
                "start": 0,
                "end": 30,
                "number_of_invoices": 1,
                "amount_paid": 6734.56,
                "amount_unpaid": 18703.86,
                "percent_of_current": 1.0
            },
            {
                "start": 31,
                "end": 60,
                "number_of_invoices": 0,
                "amount_paid": 0,
                "amount_unpaid": 0,
                "percent_of_current": 0.0
```

NAV ☰


```
            },
            {
                "start": 61,
                "end": 90,
                "number_of_invoices": 0,
                "amount_paid": 0,
                "amount_unpaid": 0,
                "percent_of_current": 0.0
            },
            {
                "start": 91,
                "end": null,
                "number_of_invoices": 0,
                "amount_paid": 0,
                "amount_unpaid": 0,
                "percent_of_current": 0.0
            }
        ],
        "overdue_periods": [
            {
                "start": 1,
                "end": 30,
                "number_of_invoices": 0,
                "amount_paid": 0,
                "amount_unpaid": 0,
                "percent_of_overdue": 0.0
            },
            {
                "start": 31,
                "end": 60,
                "number_of_invoices": 1,
                "amount_paid": 4691.28,
                "amount_unpaid": 13201.93,
                "percent_of_overdue": 1.0
            },
            {
                "start": 61,
                "end": 90,
                "number_of_invoices": 0,
                "amount_paid": 0,
                "amount_unpaid": 0,
                "percent_of_overdue": 0.0
            },
            {
                "start": 91,
                "end": null,
```

NAV ☰


```
                "number_of_invoices": 0,
                "amount_paid": 0,
                "amount_unpaid": 0,
                "percent_of_overdue": 0.0
            }
        ],
        "top_overdue_invoices": [
            {
                "invoice_id": "d0f72049-61f7-4d2c-8799-05f2fea8ac1a",
                "invoice_number": "973458",
                "invoice_date": "2020-06-24",
                "invoice_due_date": "2020-07-07",
                "days_overdue": 55,
                "amount_invoiced": 17893.21,
                "amount_paid": 4691.28,
                "amount_unpaid": 13201.93,
                "percent_of_overdue": 1.0
            }
        ],
        "top_overdue_suppliers": [
            {
                "contact_id": "89b8d79c-7189-451b-bc18-fa17338c0ae7",
                "company_name": "Vortex Solar",
                "number_of_invoices": 1,
                "amount_paid": 4691.28,
                "amount_unpaid": 13201.93,
                "overdue_periods": [
                    {
                        "start": 1,
                        "end": 30,
                        "number_of_invoices": 0,
                        "amount_paid": 0,
                        "amount_unpaid": 0,
                        "percent_of_overdue": 0.0
                    },
                    {
                        "start": 31,
                        "end": 60,
                        "number_of_invoices": 1,
                        "amount_paid": 4691.28,
                        "amount_unpaid": 13201.93,
                        "percent_of_overdue": 1.0
                    },
                    {
                        "start": 61,
                        "end": 90,
```

NAV


```
                    "number_of_invoices": 0,
                    "amount_paid": 0,
                    "amount_unpaid": 0,
                    "percent_of_overdue": 0.0
                },
                {
                    "start": 91,
                    "end": null,
                    "number_of_invoices": 0,
                    "amount_paid": 0,
                    "amount_unpaid": 0,
                    "percent_of_overdue": 0.0
                }
            ]
        }
    ],
    "top_paid_invoices": [
        {
            "invoice_id": "33989e4c-168f-4ecc-9c75-0c67e510d36e",
            "invoice_number": "9346810",
            "invoice_date": "2020-04-13",
            "amount_paid": 5000.0,
            "percent_of_paid": 1.0
        }
    ],
    "top_paid_suppliers": [
        {
            "contact_id": "89b8d79c-7189-451b-bc18-fa17338c0ae7",
            "company_name": "Vortex Solar",
            "number_of_invoices": 3,
            "amount_paid": 16425.84
        }
    ],
    "history": [
        {
            "period_start": "2020-01-01",
            "period_end": "2020-01-31",
            "amount_invoiced": 0.0,
            "amount_paid": 0.0
        },
        {
            "period_start": "2020-02-01",
            "period_end": "2020-02-29",
            "amount_invoiced": 0.0,
            "amount_paid": 0.0
        },
```

NAV


```
            {
                "period_start": "2020-03-01",
                "period_end": "2020-03-31",
                "amount_invoiced": 0.0,
                "amount_paid": 0.0
            },
            {
                "period_start": "2020-04-01",
                "period_end": "2020-04-30",
                "amount_invoiced": 5000.0,
                "amount_paid": 0.0
            },
            {
                "period_start": "2020-05-01",
                "period_end": "2020-05-31",
                "amount_invoiced": 0.0,
                "amount_paid": 0.0
            },
            {
                "period_start": "2020-06-01",
                "period_end": "2020-06-30",
                "amount_invoiced": 17893.21,
                "amount_paid": 0.0
            },
            {
                "period_start": "2020-07-01",
                "period_end": "2020-07-31",
                "amount_invoiced": 25438.42,
                "amount_paid": 11425.84
            },
            {
                "period_start": "2020-08-01",
                "period_end": "2020-08-31",
                "amount_invoiced": 0.0,
                "amount_paid": 0.0
            }
        ],
        "outstanding_invoices": [
            {
                "invoice_id": "165c8134-1105-43ec-a070-d8a68a1b5667",
                "invoice_number": "6843265",
                "invoice_date": "2020-07-12",
                "invoice_due_date": "2020-09-20",
                "days_until_due": 20,
                "days_overdue": null,
                "is_overdue": false,
```

NAV


```
                "number_of_payments": 1,
                "latest_payment_date": "2020-07-25",
                "amount_invoiced": 25438.42,
                "amount_paid": 6734.56,
                "amount_unpaid": 18703.86
            },
            {
                "invoice_id": "d0f72049-61f7-4d2c-8799-05f2fea8ac1a",
                "invoice_number": "973458",
                "invoice_date": "2020-06-24",
                "invoice_due_date": "2020-07-07",
                "days_until_due": null,
                "days_overdue": 55,
                "is_overdue": true,
                "number_of_payments": 1,
                "latest_payment_date": "2020-07-01",
                "amount_invoiced": 17893.21,
                "amount_paid": 4691.28,
                "amount_unpaid": 13201.93
            }
        ]
    }
}
```


RESPONSE

Field	Description
`currency_code`	Currency code associated with monetary response values.
`analysis_start`	Start date of the analysis period. If `start_date` was provided in the request, it is reflected here. If `start_date` was not provided, this value is derived dynamically.
`analysis_end`	End date of the analysis period. If `end_date` was provided in the request, it is reflected here. If `end_date` was not provided, this value is derived dynamically.
`receivable`	Invoice data for receivable invoices from customers.
`total_number_of_invoices`	Total number of invoices issued in the analysis period.
`total_amount_invoiced`	Total amount invoiced in the analysis period.

NAV

Field	Description
total_amount_paid	Total amount paid in the analysis period.
total_amount_unpaid	Total amount unpaid in the analysis period.
paid_summary	Summary of invoices that are paid in full.
number_of_invoices	Number of paid invoices.
amount_paid	Amount paid on paid invoices.
outstanding_summary	Summary of invoices that are outstanding.
number_of_invoices	Number of outstanding invoices.
amount_paid	Amount paid on outstanding invoices.
amount_unpaid	Amount unpaid on outstanding invoices.
current_summary	Summary of outstanding invoices that are current.
number_of_invoices	Number of current invoices.
amount_paid	Amount paid on current invoices.
amount_unpaid	Amount unpaid on current invoices.
percent_of_outstanding	Amount unpaid on current invoices as a percentage of amount unpaid on all outstanding invoices.
overdue_summary	Summary of outstanding invoices that are overdue.
number_of_invoices	Number of overdue invoices.
amount_paid	Amount paid on overdue invoices.
amount_unpaid	Amount unpaid on overdue invoices.
percent_of_outstanding	Amount unpaid on overdue invoices as a percentage of amount unpaid on all outstanding invoices.
future_due_periods	Future periods representing the number of days until current invoices will become due, as of the analysis end date. If

NAV

Field	Description
	`future_due_periods` is not provided in the request, 30-day intervals will be returned. Each entry has the fields shown below.
`start`	Start day of the period.
`end`	End day of the period.
`number_of_invoices`	Number of current invoices that will become due in the period.
`amount_paid`	Amount paid on current invoices that will become due in the period.
`amount_unpaid`	Amount unpaid on current invoices that will become due in the period.
`percent_of_current`	Amount unpaid on current invoices that will become due in the period as a percentage of amount unpaid on all current invoices.
`overdue_periods`	Periods representing the number of days that overdue invoices have been past due, as of the analysis end date. If `overdue_periods` is not provided in the request, 30-day intervals will be returned. Each entry has the fields shown below.
`start`	Start day of the period.
`end`	End day of the period.
`number_of_invoices`	Number of overdue invoices that have been past due for a number of days within the period.
`amount_paid`	Amount paid on overdue invoices that have been past due for a number of days within the period.
`amount_unpaid`	Amount unpaid on overdue invoices that have been past due for a number of days within the period.
`percent_of_overdue`	Amount unpaid on overdue invoices that have been past due for a number of days within the period as a percentage of the amount unpaid on all overdue invoices.
`top_overdue_invoices`	List of the top overdue invoices. Returns up to a maximum number of records specified in `response_limit`. Each entry has the fields shown below.

NAV

Field	Description
`invoice_id`	ID of the Nucleus Invoice.
`invoice_number`	Number of the invoice.
`invoice_date`	Date the invoice was issued.
`invoice_due_date`	Due date of the invoice.
`days_overdue`	Number of days the invoice has been overdue.
`amount_invoiced`	Amount of the invoice.
`amount_paid`	Amount paid on the invoice.
`amount_unpaid`	Amount unpaid on the invoice.
`percent_of_overdue`	Amount unpaid on the invoice as a percentage of the amount unpaid on all overdue invoices.
`top_overdue_customers`	List of the top overdue customers. Returns up to a maximum number of records specified in `response_limit`. Each entry has the fields shown below.
`contact_id`	ID of the Nucleus Contact that represents the customer.
`company_name`	Company name for the customer.
`number_of_invoices`	Number of the customer's overdue invoices.
`amount_paid`	Amount paid on the customer's overdue invoices.
`amount_unpaid`	Amount unpaid on the customer's overdue invoices.
`overdue_periods`	Periods representing the number of days that the customer's overdue invoices have been past due. If `overdue_periods` is not provided in the request, 30-day intervals will be returned. Each entry has the fields shown below.
`start`	Start day of the period.
`end`	End day of the period.

NAV

Field	Description
`number_of_invoices`	Number of the customer's overdue invoices that have been past due for a number of days within the period.
`amount_paid`	Amount paid on the customer's overdue invoices that have been past due for a number of days within the period.
`amount_unpaid`	Amount unpaid on the customer's overdue invoices that have been past due for a number of days within the period.
`percent_of_overdue`	Amount unpaid on the customer's overdue invoices that have been past due for a number of days within the period as a percentage of the amount unpaid on all of the customer's overdue invoices.
`top_paid_invoices`	List of the top paid invoices. Returns up to a maximum number of records specified in `response_limit`. Each entry has the fields shown below.
`invoice_id`	ID of the Nucleus Invoice.
`invoice_number`	Number of the invoice.
`invoice_date`	Date the invoice was issued.
`amount_paid`	Amount paid on the invoice.
`percent_of_paid`	Amount paid on the invoice as a percentage of the amount paid on all paid invoices.
`top_paid_customers`	List of the top paid customers. Returns up to a maximum number of records specified in `response_limit`. Each entry has the fields shown below.
`contact_id`	ID of the Nucleus Contact that represents the customer.
`company_name`	Company name for the customer.
`number_of_invoices`	Number of the customer's paid invoices.
`amount_paid`	Amount paid on the customer's paid invoices.
`history`	Historical invoice data, with frequency dependent on the `history_frequency_interval` value passed in the request. Each entry has the fields shown below.

NAV

Field	Description
`period_start`	Period start date.
`period_end`	Period end date.
`amount_invoiced`	Amount invoiced in the period.
`amount_paid`	Amount paid in the period.
`outstanding_invoices`	Each entry has the fields shown below.
`invoice_id`	ID of the Nucleus Invoice.
`invoice_number`	Number of the invoice.
`invoice_date`	Date the invoice was issued.
`invoice_due_date`	Due date of the invoice.
`days_until_due`	As of the analysis end date, the number of days until the invoice due date. Will be `null` if `is_overdue` is `true`.
`days_overdue`	As of the analysis end date, the number of days that the invoice has been outstanding since the invoice due date. Will be `null` if `is_overdue` is `false`.
`is_overdue`	If `true`, the invoice is overdue. If `false`, the invoice is current.
`number_of_payments`	Number of payments made on the invoice.
`latest_payment_date`	Latest date a payment was made on the invoice.
`amount_invoiced`	Amount of the invoice.
`amount_paid`	Amount paid on the invoice.
`amount_unpaid`	Amount unpaid on the invoice.
`payable`	Invoice data for payable invoices from suppliers.
`total_number_of_invoices`	Total number of invoices issued in the analysis period.
`total_amount_invoiced`	Total amount invoiced in the analysis period.

NAV

Field	Description
total_amount_paid	Total amount paid in the analysis period.
total_amount_unpaid	Total amount unpaid in the analysis period.
paid_summary	Summary of invoices that are paid in full.
number_of_invoices	Number of paid invoices.
amount_paid	Amount paid on paid invoices.
outstanding_summary	Summary of invoices that are outstanding.
number_of_invoices	Number of outstanding invoices.
amount_paid	Amount paid on outstanding invoices.
amount_unpaid	Amount unpaid on outstanding invoices.
current_summary	Summary of outstanding invoices that are current.
number_of_invoices	Number of current invoices.
amount_paid	Amount paid on current invoices.
amount_unpaid	Amount unpaid on current invoices.
percent_of_outstanding	Amount unpaid on current invoices as a percentage of amount unpaid on all outstanding invoices.
overdue_summary	Summary of outstanding invoices that are overdue.
number_of_invoices	Number of overdue invoices.
amount_paid	Amount paid on overdue invoices.
amount_unpaid	Amount unpaid on overdue invoices.
percent_of_outstanding	Amount unpaid on overdue invoices as a percentage of amount unpaid on all outstanding invoices.
future_due_periods	Future periods representing the number of days until current invoices will become due, as of the analysis end date. If

NAV

Field	Description
	`future_due_periods` is not provided in the request, 30-day intervals will be returned. Each entry has the fields shown below.
`start`	Start day of the period.
`end`	End day of the period.
`number_of_invoices`	Number of current invoices that will become due in the period.
`amount_paid`	Amount paid on current invoices that will become due in the period.
`amount_unpaid`	Amount unpaid on current invoices that will become due in the period.
`percent_of_current`	Amount unpaid on current invoices that will become due in the period as a percentage of amount unpaid on all current invoices.
`overdue_periods`	Periods representing the number of days that overdue invoices have been past due, as of the analysis end date. If `overdue_periods` is not provided in the request, 30-day intervals will be returned. Each entry has the fields shown below.
`start`	Start day of the period.
`end`	End day of the period.
`number_of_invoices`	Number of overdue invoices that have been past due for a number of days within the period.
`amount_paid`	Amount paid on overdue invoices that have been past due for a number of days within the period.
`amount_unpaid`	Amount unpaid on overdue invoices that have been past due for a number of days within the period.
`percent_of_overdue`	Amount unpaid on overdue invoices that have been past due for a number of days within the period as a percentage of the amount unpaid on all overdue invoices.
`top_overdue_invoices`	List of the top overdue invoices. Returns up to a maximum number of records specified in `response_limit`. Each entry has the fields shown below.

NAV

Field	Description
`invoice_id`	ID of the Nucleus Invoice.
`invoice_number`	Number of the invoice.
`invoice_date`	Date the invoice was issued.
`invoice_due_date`	Due date of the invoice.
`days_overdue`	Number of days the invoice has been overdue.
`amount_invoiced`	Amount of the invoice.
`amount_paid`	Amount paid on the invoice.
`amount_unpaid`	Amount unpaid on the invoice.
`percent_of_overdue`	Amount unpaid on the invoice as a percentage of the amount unpaid on all overdue invoices.
`top_overdue_suppliers`	List of the top overdue suppliers. Returns up to a maximum number of records specified in `response_limit`. Each entry has the fields shown below.
`contact_id`	ID of the Nucleus Contact that represents the supplier.
`company_name`	Company name for the supplier.
`number_of_invoices`	Number of the supplier's overdue invoices.
`amount_paid`	Amount paid on the supplier's overdue invoices.
`amount_unpaid`	Amount unpaid on the supplier's overdue invoices.
`overdue_periods`	Periods representing the number of days that the supplier's overdue invoices have been past due. If `overdue_periods` is not provided in the request, 30-day intervals will be returned. Each entry has the fields shown below.
`start`	Start day of the period.
`end`	End day of the period.

NAV

Field	Description
`number_of_invoices`	Number of the supplier's overdue invoices that have been past due for a number of days within the period.
`amount_paid`	Amount paid on the supplier's overdue invoices that have been past due for a number of days within the period.
`amount_unpaid`	Amount unpaid on the supplier's overdue invoices that have been past due for a number of days within the period.
`percent_of_overdue`	Amount unpaid on the supplier's overdue invoices that have been past due for a number of days within the period as a percentage of the amount unpaid on all of the supplier's overdue invoices.
`top_paid_invoices`	List of the top paid invoices. Returns up to a maximum number of records specified in `response_limit`. Each entry has the fields shown below.
`invoice_id`	ID of the Nucleus Invoice.
`invoice_number`	Number of the invoice.
`invoice_date`	Date the invoice was issued.
`amount_paid`	Amount paid on the invoice.
`percent_of_paid`	Amount paid on the invoice as a percentage of the amount paid on all paid invoices.
`top_paid_suppliers`	List of the top paid suppliers. Returns up to a maximum number of records specified in `response_limit`. Each entry has the fields shown below.
`contact_id`	ID of the Nucleus Contact that represents the supplier.
`company_name`	Company name for the supplier.
`number_of_invoices`	Number of the supplier's paid invoices.
`amount_paid`	Amount paid on the supplier's paid invoices.
`history`	Historical invoice data, with frequency dependent on the `history_frequency_interval` value passed in the request. Each entry has the fields shown below.

NAV

Field	Description
`period_start`	Period start date.
`period_end`	Period end date.
`amount_invoiced`	Amount invoiced in the period.
`amount_paid`	Amount paid in the period.
`outstanding_invoices`	Each entry has the fields shown below.
`invoice_id`	ID of the Nucleus Invoice.
`invoice_number`	Number of the invoice.
`invoice_date`	Date the invoice was issued.
`invoice_due_date`	Due date of the invoice.
`days_until_due`	As of the analysis end date, the number of days until the invoice due date. Will be `null` if `is_overdue` is `true`.
`days_overdue`	As of the analysis end date, the number of days that the invoice has been outstanding since the invoice due date. Will be `null` if `is_overdue` is `false`.
`is_overdue`	If `true`, the invoice is overdue. If `false`, the invoice is current.
`number_of_payments`	Number of payments made on the invoice.
`latest_payment_date`	Latest date a payment was made on the invoice.
`amount_invoiced`	Amount of the invoice.
`amount_paid`	Amount paid on the invoice.
`amount_unpaid`	Amount unpaid on the invoice.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Contact`
- `Invoice`

NAV
- Invoice Payment

Decision Tree Result

Returns the `model_id` or `allocation_id` by traversing the decision tree and finding the corresponding leaf node for the `answer_ids` provided in the `client_response`.

HTTP REQUEST

`POST /decision_tree_result`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "entity_type": "model",
        "answers": [
            {
                "answer_id": "2fcec30e-fad1-4845-a507-100c5deaeef3"
            }
    ]
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/decision_tree_result"
```

ARGUMENTS

Parameter	Description
`entity_type` *string*	The entity to which the answers correspond. The value can be `model` or `allocation`.
`answers` *array[map]*	An array of objects, each with an answer provided in relation to the entity.
`answer_id` *UUID*	The ID of the answer being provided.

NAV

Example Response

```
{
    "entity_type": "model",
    "entity_id": "33fd99ce-3d28-46d7-a727-46b1b694fac1"
}
```

RESPONSE

Field	Description
`entity_type`	The entity to which the answers correspond. The value can be `model_id` or `allocation_id`.
`entity_id`	The ID of the entity, as a result of systemically traversing the decision tree.

NUCLEUS DATA DEPENDENCIES

- `Allocation`
- `Model`
- `Decision Tree`
- `Node`
- `Node Relationship`
- `Client`
- `Client Response`
- `Answer`

Financial Planning

Annuity Calculator

When planning for retirement or other financial outcomes, a fixed annuity allows an investor to receive a series of payments over a given period of time in the future. This tool analyzes fixed annuities and solves for a variety of relevant variables including the annuity amount, the investor's initial balance, ongoing contributions to the annuity, the accumulation horizon, and the decumulation horizon.

Annuity Amount

NAV

This service helps determine the annuity amount that is attainable based on the details provided.

HTTP REQUEST

```
POST /annuity_calculator/annuity_amount
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "portfolio_return": 0.1,
        "initial_balance": 2000,
        "accumulation_horizon": 3,
        "decumulation_horizon": 2,
        "annuity_frequency_interval": "year",
        "inflation_rate": 0.02,
        "tax_rate": 0.25,
        "deposit_schedule": {
            "deposit_amount": 500,
            "deposit_frequency_interval": "year",
            "adjust_deposit_for_inflation": true
        }
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/annuity_calculator/annui
```

ARGUMENTS

Parameter	Description
`portfolio_return` *float, required*	The annualized rate of return of the portfolio.
`accumulation_horizon` *integer, required*	The number of years until the commencement of annuity payments.
`decumulation_horizon` *integer, required*	The number of years that annuity payments last.
`annuity_frequency_interval` *string*	The period interval that relates to `annuity_amount`. Must be one of `year`, `quarter`, `month`, `week`, or `day`. Defaults to `year`.

NAV

Parameter	Description
`initial_balance` *float*	The starting balance in the annuity plan, prior to any periodic contributions. Defaults to `0`.
`inflation_rate` *float*	The annualized inflation rate to apply to `portfolio_return`. Defaults to `0`.
`tax_rate` *float*	The tax rate to apply to `annuity_amount`. Defaults to `0`.
`deposit_schedule` *map*	The schedule of ongoing deposits to be made during the `accumulation_horizon`. Includes the fields shown below.
`deposit_amount` *float*	The periodic value of the deposit. Must be greater than or equal to 0. Defaults to `0`.
`deposit_frequency_interval` *string*	The unit of time associated with deposit_amount. Must be one of `year`, `quarter`, `month`, `week`, or `day`. Defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, `deposit_amount` will be adjusted over time based on `inflation_rate`. Defaults to `true`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

Example Response

```
{
    "annuity_amount": 1753.96,
    "annuity_frequency_interval": "year",
    "total_earnings": 1482.95,
    "total_contributions": 1530.2,
    "cumulative_annuity_amount": 5013.15,
```

NAV


```
        "total_taxes": 1671.05,
        "return_details": {
            "0": {
                "period_earnings": 0,
                "period_contribution": 0,
                "period_withdrawal": 0,
                "period_tax": 0,
                "cumulative_earnings": 0,
                "cumulative_contributions": 0,
                "cumulative_withdrawals": 0,
                "cumulative_tax": 0,
                "ending_balance": 2000
            },
            "1": {
                "period_earnings": 200,
                "period_contribution": 500,
                "period_withdrawal": 0,
                "period_tax": 0,
                "cumulative_earnings": 200,
                "cumulative_contributions": 500,
                "cumulative_withdrawals": 0,
                "cumulative_tax": 0,
                "ending_balance": 2700
            },
            "2": {
                "period_earnings": 270,
                "period_contribution": 510,
                "period_withdrawal": 0,
                "period_tax": 0,
                "cumulative_earnings": 470,
                "cumulative_contributions": 1010,
                "cumulative_withdrawals": 0,
                "cumulative_tax": 0,
                "ending_balance": 3480
            },
            "3": {
                "period_earnings": 348,
                "period_contribution": 520.2,
                "period_withdrawal": 0,
                "period_tax": 0,
                "cumulative_earnings": 818,
                "cumulative_contributions": 1530.2,
                "cumulative_withdrawals": 0,
                "cumulative_tax": 0,
                "ending_balance": 4348.2
            },
```

NAV


```
            "4": {
                "period_earnings": 434.82,
                "period_contribution": 0,
                "period_withdrawal": 2481.76,
                "period_tax": 827.25,
                "cumulative_earnings": 1252.82,
                "cumulative_contributions": 1530.2,
                "cumulative_withdrawals": 2481.76,
                "cumulative_tax": 827.25,
                "ending_balance": 2301.26
            },
            "5": {
                "period_earnings": 230.13,
                "period_contribution": 0,
                "period_withdrawal": 2531.39,
                "period_tax": 843.8,
                "cumulative_earnings": 1482.95,
                "cumulative_contributions": 1530.2,
                "cumulative_withdrawals": 5013.15,
                "cumulative_tax": 1671.05,
                "ending_balance": 0
            }
        }
    }
```


RESPONSE

Field	Description
`annuity_amount`	The periodic annuity amount.
`annuity_frequency_interval`	The frequency at which `annuity_amount` is drawn from the portfolio.
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`cumulative_annuity_amount`	The total amount received from the annuity over the course of the plan.
`total_taxes`	The total taxes paid on annuity payments over decumulation_horizon.

NAV

Field	Description
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions are made.
`period_contribution`	The deposit made for this period.
`period_withdrawal`	The withdrawal made for this period.
`period_tax`	The tax amount for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`cumulative_tax`	The cumulative taxes up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

Initial Balance

NAV

This service helps determine the initial balance that an investor needs in order to satisfy the annuity details provided.

HTTP REQUEST

```
POST /annuity_calculator/initial_balance
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "portfolio_return": 0.1,
        "accumulation_horizon": 3,
        "decumulation_horizon": 2,
        "annuity_amount": 3000,
        "annuity_frequency_interval": "year",
        "inflation_rate": 0.02,
        "tax_rate": 0.25,
        "deposit_schedule": {
            "deposit_amount": 500,
            "deposit_frequency_interval": "year",
            "adjust_deposit_for_inflation": true
        }
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/annuity_calculator/initi
```

ARGUMENTS

Parameter	Description
`portfolio_return` *float, required*	The annualized rate of return of the portfolio.
`accumulation_horizon` *integer, required*	The number of years until the commencement of annuity payments.
`decumulation_horizon` *integer, required*	The number of years that annuity payments last.
`annuity_amount` *float, conditional requirement*	The amount of the annuity, represented in today's dollars.

NAV

Parameter	Description
`annuity_frequency_interval` *string*	The period interval that relates to `annuity_amount`. Must be one of `year`, `quarter`, `month`, `week`, or `day`. Defaults to `year`.
`inflation_rate` *float*	The annualized inflation rate to apply to `portfolio_return`. Defaults to `0`.
`tax_rate` *float*	The tax rate to apply to `annuity_amount`. Defaults to `0`.
`deposit_schedule` *map*	The schedule of ongoing deposits to be made during the `accumulation_horizon`. Includes the fields shown below.
`deposit_amount` *float*	The periodic value of the deposit. Must be greater than or equal to 0. Defaults to `0`.
`deposit_frequency_interval` *string*	The unit of time associated with deposit_amount. Must be one of `year`, `quarter`, `month`, `week`, or `day`. Defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, `deposit_amount` will be adjusted over time based on `inflation_rate`. Defaults to `true`.

Example Response


```
{
    "initial_balance": 4320.83,
    "total_earnings": 2723.53,
    "total_contributions": 1530.2,
    "cumulative_annuity_amount": 8574.56,
    "total_taxes": 2858.19,
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_contribution": 0,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 0,
            "cumulative_contributions": 0,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
```

NAV III


```
            "ending_balance": 4320.83
        },
        "1": {
            "period_earnings": 432.08,
            "period_contribution": 500,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 432.08,
            "cumulative_contributions": 500,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 5252.91
        },
        "2": {
            "period_earnings": 525.29,
            "period_contribution": 510,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 957.37,
            "cumulative_contributions": 1010,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 6288.21
        },
        "3": {
            "period_earnings": 628.82,
            "period_contribution": 520.2,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 1586.2,
            "cumulative_contributions": 1530.2,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 7437.23
        },
        "4": {
            "period_earnings": 743.72,
            "period_contribution": 0,
            "period_withdrawal": 4244.83,
            "period_tax": 1414.94,
            "cumulative_earnings": 2329.92,
            "cumulative_contributions": 1530.2,
            "cumulative_withdrawals": 4244.83,
            "cumulative_tax": 1414.94,
            "ending_balance": 3936.12
        },
```

NAV


```
            "5": {
                "period_earnings": 393.61,
                "period_contribution": 0,
                "period_withdrawal": 4329.73,
                "period_tax": 1443.24,
                "cumulative_earnings": 2723.53,
                "cumulative_contributions": 1530.2,
                "cumulative_withdrawals": 8574.56,
                "cumulative_tax": 2858.19,
                "ending_balance": 0
            }
        }
    }
```


RESPONSE

Field	Description
`initial_balance`	The starting balance in the annuity plan, prior to any periodic contributions.
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`cumulative_annuity_amount`	The total amount received from the annuity over the course of the plan.
`total_taxes`	The total taxes paid on annuity payments over decumulation_horizon.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions are made.
`period_contribution`	The deposit made for this period.
`period_withdrawal`	The withdrawal made for this period.

NAV

Field	Description
`period_tax`	The tax amount for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`cumulative_tax`	The cumulative taxes up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

Deposit Amount

This service helps determine the recurring amount to be contributed over the accumulation horizon, in order to satisfy the annuity details provided.

HTTP REQUEST

`POST /annuity_calculator/deposit_amount`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "portfolio_return": 0.1,
        "initial_balance": 2000,
        "accumulation_horizon": 3,
        "decumulation_horizon": 2,
        "annuity_amount": 3000,
        "annuity_frequency_interval": "year",
        "inflation_rate": 0.02,
        "tax_rate": 0.25,
```

NAV

```
        "deposit_schedule": {
            "deposit_frequency_interval": "year",
            "adjust_deposit_for_inflation": true
        }
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/annuity_calculator/depos
```

ARGUMENTS

Parameter	Description
`portfolio_return` *float, required*	The annualized rate of return of the portfolio.
`accumulation_horizon` *integer, required*	The number of years until the commencement of annuity payments.
`decumulation_horizon` *integer, required*	The number of years that annuity payments last.
`annuity_amount` *float, required*	The amount of the annuity, represented in today's dollars.
`annuity_frequency_interval` *string*	The period interval that relates to `annuity_amount`. Must be one of `year`, `quarter`, `month`, `week`, or `day`. Defaults to `year`.
`initial_balance` *float*	The starting balance in the annuity plan, prior to any periodic contributions. Defaults to `0`.
`inflation_rate` *float*	The annualized inflation rate to apply to `portfolio_return`. Defaults to `0`.
`tax_rate` *float*	The tax rate to apply to `annuity_amount`. Defaults to `0`.
`deposit_schedule` *map*	The schedule of deposits to be made during the `accumulation_horizon`. Includes the fields shown below.
`deposit_frequency_interval` *string*	The period interval to be used in relation to the `deposit_amount`. Must be one of `year`, `quarter`, `month`, `week`, or `day`. Defaults to `year`.

NAV

Parameter	Description
`adjust_deposit_for_inflation` *boolean*	If `true`, the `deposit_amount` will be increased by `inflation_rate`. Defaults to `true`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

Example Response


```
{
    "deposit_amount": 1415.97,
    "deposit_frequency_interval": "year",
    "total_earnings": 2241.12,
    "total_contributions": 4333.44,
    "cumulative_annuity_amount": 8574.56,
    "total_taxes": 2858.19,
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_contribution": 0,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 0,
            "cumulative_contributions": 0,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 2000
        },
        "1": {
            "period_earnings": 200,
            "period_contribution": 1415.97,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 200,
```

NAV III


```
            "cumulative_contributions": 1415.97,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 3615.97
        },
        "2": {
            "period_earnings": 361.6,
            "period_contribution": 1444.29,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 561.6,
            "cumulative_contributions": 2860.26,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 5421.86
        },
        "3": {
            "period_earnings": 542.19,
            "period_contribution": 1473.18,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 1103.78,
            "cumulative_contributions": 4333.44,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 7437.23
        },
        "4": {
            "period_earnings": 743.72,
            "period_contribution": 0,
            "period_withdrawal": 4244.83,
            "period_tax": 1414.94,
            "cumulative_earnings": 1847.51,
            "cumulative_contributions": 4333.44,
            "cumulative_withdrawals": 4244.83,
            "cumulative_tax": 1414.94,
            "ending_balance": 3936.12
        },
        "5": {
            "period_earnings": 393.61,
            "period_contribution": 0,
            "period_withdrawal": 4329.73,
            "period_tax": 1443.24,
            "cumulative_earnings": 2241.12,
            "cumulative_contributions": 4333.44,
            "cumulative_withdrawals": 8574.56,
```

NAV

```
                "cumulative_tax": 2858.19,
                "ending_balance": 0
            }
        }
    }
```

RESPONSE

Field	Description
`deposit_amount`	The amount to be deposited per period.
`deposit_frequency_interval`	The period interval to be used in relation to the `deposit_amount`. Defaults to `year`.
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`cumulative_annuity_amount`	The total amount received from the annuity over the course of the plan.
`total_taxes`	The total taxes paid on annuity payments over decumulation_horizon.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions are made.
`period_contribution`	The deposit made for this period.
`period_withdrawal`	The withdrawal made for this period.
`period_tax`	The tax amount for this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.

NAV

Field	Description
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`cumulative_tax`	The cumulative taxes up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

Decumulation Horizon

This service helps determine the time horizon over which an annuity can persist, based on the details provided.

HTTP REQUEST

`POST /annuity_calculator/decumulation_horizon`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "portfolio_return": 0.1,
        "initial_balance": 2000,
        "accumulation_horizon": 3,
```

NAV

```
        "annuity_amount": 3000,
        "annuity_frequency_interval": "year",
        "inflation_rate": 0.02,
        "tax_rate": 0.25,
        "deposit_schedule": {
            "deposit_amount": 500,
            "deposit_frequency_interval": "year",
            "adjust_deposit_for_inflation": true
        }
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/annuity_calculator/decum
```

ARGUMENTS

Parameter	Description
`portfolio_return` *float, required*	The annualized rate of return of the portfolio.
`accumulation_horizon` *integer, required*	The number of years until the commencement of annuity payments.
`annuity_amount` *float, required*	The amount of the annuity, represented in today's dollars.
`annuity_frequency_interval` *string*	The period interval that relates to `annuity_amount`. Must be one of `year`, `quarter`, `month`, `week`, or `day`. Defaults to `year`.
`initial_balance` *float*	The starting balance in the annuity plan, prior to any periodic contributions. Defaults to `0`.
`inflation_rate` *float*	The annualized inflation rate to apply to `portfolio_return`. Defaults to `0`.
`tax_rate` *float*	The tax rate to apply to `annuity_amount`. Defaults to `0`.
`deposit_schedule` *map*	The schedule of ongoing deposits to be made during the `accumulation_horizon`. Includes the fields shown below.
`deposit_amount` *float*	The periodic value of the deposit. Must be greater than or equal to 0. Defaults to `0`.

NAV

Parameter	Description
`deposit_frequency_interval` *string*	The unit of time associated with deposit_amount. Must be one of `year`, `quarter`, `month`, `week`, or `day`. Defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, `deposit_amount` will be adjusted over time based on `inflation_rate`. Defaults to `true`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

Example Response


```
{
    "decumulation_horizon": {
        "years": 1,
        "months": 1,
        "days": 12
    },
    "total_earnings": 1259.94,
    "total_contributions": 1530.2,
    "cumulative_annuity_amount": 4790.14,
    "total_taxes": 1596.71,
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_contribution": 0,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 0,
            "cumulative_contributions": 0,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 2000
        },
```

NAV


```
        "1": {
            "period_earnings": 200,
            "period_contribution": 500,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 200,
            "cumulative_contributions": 500,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 2700
        },
        "2": {
            "period_earnings": 270,
            "period_contribution": 510,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 470,
            "cumulative_contributions": 1010,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 3480
        },
        "3": {
            "period_earnings": 348,
            "period_contribution": 520.2,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 818,
            "cumulative_contributions": 1530.2,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 4348.2
        },
        "4": {
            "period_earnings": 434.82,
            "period_contribution": 0,
            "period_withdrawal": 4244.83,
            "period_tax": 1414.94,
            "cumulative_earnings": 1252.82,
            "cumulative_contributions": 1530.2,
            "cumulative_withdrawals": 4244.83,
            "cumulative_tax": 1414.94,
            "ending_balance": 538.19
        },
        "5": {
            "period_earnings": 7.12,
```

NAV


```
                "period_contribution": 0,
                "period_withdrawal": 545.31,
                "period_tax": 181.77,
                "cumulative_earnings": 1259.94,
                "cumulative_contributions": 1530.2,
                "cumulative_withdrawals": 4790.14,
                "cumulative_tax": 1596.71,
                "ending_balance": 0
            }
        }
    }
```


RESPONSE

Field	Description
`decumulation_horizon`	The time horizon over which annuity payments persist.
`years`	The number of years in the decumulation horizon.
`months`	The number of months in the decumulation horizon.
`days`	The number of days in the decumulation horizon.
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`cumulative_annuity_amount`	The total amount received from the annuity over the course of the plan.
`total_taxes`	The total taxes paid on annuity payments over decumulation_horizon.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions are made.

NAV

Field	Description
`period_contribution`	The deposit made for this period.
`period_withdrawal`	The withdrawal made for this period.
`period_tax`	The tax amount for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`cumulative_tax`	The cumulative taxes up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

Accumulation Horizon

This service helps determine the accumulation horizon (i.e. the time period until annuity payments begin) that is needed in order to satisfy the annuity details provided.

HTTP REQUEST

```
POST /annuity_calculator/accumulation_horizon
```

NAV

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "portfolio_return": 0.1,
        "initial_balance": 2000,
        "decumulation_horizon": 2,
        "annuity_amount": 3000,
        "annuity_frequency_interval": "year",
        "inflation_rate": 0.02,
        "tax_rate": 0.25,
        "deposit_schedule": {
            "deposit_amount": 500,
            "deposit_frequency_interval": "year",
            "adjust_deposit_for_inflation": true
        }
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/annuity_calculator/accum
```

ARGUMENTS

Parameter	Description
`portfolio_return` *float, required*	The annualized rate of return of the portfolio.
`decumulation_horizon` *integer, required*	The number of years that annuity payments last.
`annuity_amount` *float, required*	The amount of the annuity, represented in today's dollars.
`annuity_frequency_interval` *string*	The period interval that relates to `annuity_amount`. Must be one of the following: `year`, `quarter`, `month`, `week`. Defaults to `year`.
`initial_balance` *float*	The starting balance in the annuity plan, prior to any periodic contributions. Defaults to `0`.
`inflation_rate` *float*	The annualized inflation rate to apply to `portfolio_return`. Defaults to `0`.

NAV

Parameter	Description
`tax_rate` *float*	The tax rate to apply to `annuity_amount`. Defaults to `0`.
`deposit_schedule` *map*	The schedule of ongoing deposits to be made during the `accumulation_horizon`. Includes the fields shown below.
`deposit_amount` *float*	The periodic value of the deposit. Must be greater than or equal to 0. Defaults to `0`.
`deposit_frequency_interval` *string*	The unit of time associated with deposit_amount. Must be one of `year`, `quarter`, `month`, `week`, or `day`. Defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, `deposit_amount` will be adjusted over time based on `inflation_rate`. Defaults to `true`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

Example Response


```
{
    "accumulation_horizon": {
        "years": 6,
        "months": 0,
        "days": 0
    },
    "total_earnings": 3632.85,
    "total_contributions": 3467.25,
    "cumulative_annuity_amount": 9100.11,
    "total_taxes": 3033.37,
    "return_details": {
        "0": {
            "period_earnings": 0,
```

NAV III


```
            "period_contribution": 0,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 0,
            "cumulative_contributions": 0,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 2000
        },
        "1": {
            "period_earnings": 200,
            "period_contribution": 500,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 200,
            "cumulative_contributions": 500,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 2700
        },
        "2": {
            "period_earnings": 270,
            "period_contribution": 510,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 470,
            "cumulative_contributions": 1010,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 3480
        },
        "3": {
            "period_earnings": 348,
            "period_contribution": 520.2,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 818,
            "cumulative_contributions": 1530.2,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 4348.2
        },
        "4": {
            "period_earnings": 434.82,
            "period_contribution": 530.6,
            "period_withdrawal": 0,
```

NAV III


```
            "period_tax": 0,
            "cumulative_earnings": 1252.82,
            "cumulative_contributions": 2060.8,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 5313.62
        },
        "5": {
            "period_earnings": 531.36,
            "period_contribution": 541.22,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 1784.18,
            "cumulative_contributions": 2602.02,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 6386.2
        },
        "6": {
            "period_earnings": 638.62,
            "period_contribution": 552.04,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 2422.8,
            "cumulative_contributions": 3154.06,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 7576.86
        },
        "7": {
            "period_earnings": 3.01,
            "period_contribution": 313.19,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 2425.81,
            "cumulative_contributions": 3467.25,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 7893.06
        },
        "8": {
            "period_earnings": 789.31,
            "period_contribution": 0,
            "period_withdrawal": 4505,
            "period_tax": 1501.67,
            "cumulative_earnings": 3215.12,
```

NAV


```
                "cumulative_contributions": 3467.25,
                "cumulative_withdrawals": 4505,
                "cumulative_tax": 1501.67,
                "ending_balance": 4177.37
            },
            "9": {
                "period_earnings": 417.74,
                "period_contribution": 0,
                "period_withdrawal": 4595.1,
                "period_tax": 1531.7,
                "cumulative_earnings": 3632.85,
                "cumulative_contributions": 3467.25,
                "cumulative_withdrawals": 9100.11,
                "cumulative_tax": 3033.37,
                "ending_balance": 0
            }
        }
    }
```


RESPONSE

Field	Description
`accumulation_horizon`	The time horizon until the commencement of annuity payments.
`years`	The number of years in the accumulation horizon.
`months`	The number of months in the accumulation horizon.
`days`	The number of days in the accumulation horizon.
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`cumulative_annuity_amount`	The total amount received from the annuity over the course of the plan.
`total_taxes`	The total taxes paid on annuity payments over decumulation_horizon.

NAV

Field	Description
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions are made.
`period_contribution`	The deposit made for this period.
`period_withdrawal`	The withdrawal made for this period.
`period_tax`	The tax amount for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`cumulative_tax`	The cumulative taxes up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

NAV ☰

Education Planning Calculator

When planning for a future education expense, users face common questions when deciding where or when to attend. This tool helps in the decision process with three different endpoints to solve for the following parameters: deposit amount, percent of costs covered, and total annual cost. With these calculators, the user will be able to determine how much needs to be saved to attend a particular school, or what schools would be affordable given their financial circumstances. For all calculators, deposits are assumed to continue through the decumulation phase.

Deposit Amount

A calculator to help plan for an education goal by showing the user the amount that needs to be deposited for a given period.

HTTP REQUEST

```
POST /education_calculator/deposit_amount
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
         "initial_balance": 5000,
         "accumulation_horizon": 3,
         "decumulation_horizon": 4,
         "total_annual_cost": 40000,
         "portfolio_return": 0.06,
         "percent_of_costs_covered": 0.78,
         "education_inflation_rate": 0.05,
         "general_inflation_rate": 0.02,
         "tax_rate": 0.33,
         "deposit_schedule": {
            "deposit_frequency_interval": "year",
            "adjust_deposit_for_inflation": true
         }
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/education_calculator/d
```

ARGUMENTS

NAV

Parameter	Description
`accumulation_horizon` *integer, required*	The amount of years until funds are needed to pay for an education expense.
`decumulation_horizon` *integer, required*	The number of years to pay the `total_annual_cost`.
`total_annual_cost` *float, required*	The total cost paid per year for the education goal, represented in today's dollars.
`portfolio_return` *float, required*	The portfolio return for the length of the `accumulation_horizon` and `decumulation_horizon`.
`percent_of_costs_covered` *float*	The percentage of `total_annual_cost` to be paid for. If excluded, defaults to `1`.
`initial_balance` *float*	The amount currently saved for the education goal. If excluded, defaults to `0`.
`education_inflation_rate` *float*	The inflation rate attributed to `total_annual_cost`. If excluded, defaults to `0.05`.
`general_inflation_rate` *float*	The rate used to increase deposits for inflation. If excluded, defaults to `0.02`.
`tax_rate` *float*	The tax rate to apply to investments upon removal from the portfolio. If excluded, defaults to `0`.
`deposit_schedule` *map*	The deposit schedule for the education goal.
`deposit_frequency_interval` *string*	The period interval to be used in relation to the `deposit_amount`. The value may be one of the following: `year`, `quarter`, `month`, or `week`. If excluded, defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, the deposits should be increased over time with the rate of inflation. If excluded, defaults to `true`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

NAV

Parameter	Description
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

Example Response


```
{
    "deposit_amount": 37564.93,
    "deposit_frequency_interval": "year",
    "projected_accumulation_savings": 127860.79,
    "total_earnings": 28506.82,
    "total_contributions": 279268.34,
    "total_cost": 209559.36,
    "total_taxes": 103215.8,
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_contribution": 0,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 0,
            "cumulative_contributions": 0,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 5000
        },
        "1": {
            "period_earnings": 300,
            "period_contribution": 37564.93,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 300,
            "cumulative_contributions": 37564.93,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 42864.93
        },
        "2": {
            "period_earnings": 2571.9,
```

NAV III


```
            "period_contribution": 38316.23,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 2871.9,
            "cumulative_contributions": 75881.16,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 83753.06
        },
        "3": {
            "period_earnings": 5025.18,
            "period_contribution": 39082.55,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 7897.08,
            "cumulative_contributions": 114963.71,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 127860.79
        },
        "4": {
            "period_earnings": 7671.65,
            "period_contribution": 39864.2,
            "period_withdrawal": -48620.25,
            "period_taxes": -23947.29,
            "cumulative_earnings": 15568.73,
            "cumulative_contributions": 154827.92,
            "cumulative_withdrawals": -48620.25,
            "cumulative_taxes": -23947.29,
            "ending_balance": 102829.11
        },
        "5": {
            "period_earnings": 6169.75,
            "period_contribution": 40661.49,
            "period_withdrawal": -51051.26,
            "period_taxes": -25144.65,
            "cumulative_earnings": 21738.47,
            "cumulative_contributions": 195489.41,
            "cumulative_withdrawals": -99671.51,
            "cumulative_taxes": -49091.94,
            "ending_balance": 73464.43
        },
        "6": {
            "period_earnings": 4407.87,
            "period_contribution": 41474.72,
            "period_withdrawal": -53603.83,
```

NAV


```
                "period_taxes": -26401.88,
                "cumulative_earnings": 26146.34,
                "cumulative_contributions": 236964.13,
                "cumulative_withdrawals": -153275.34,
                "cumulative_taxes": -75493.82,
                "ending_balance": 39341.3
            },
            "7": {
                "period_earnings": 2360.48,
                "period_contribution": 42304.21,
                "period_withdrawal": -56284.02,
                "period_taxes": -27721.98,
                "cumulative_earnings": 28506.82,
                "cumulative_contributions": 279268.34,
                "cumulative_withdrawals": -209559.36,
                "cumulative_taxes": -103215.8,
                "ending_balance": 0
            }
        }
    }
```


RESPONSE

Field	Description
`deposit_amount`	The deposit amount to meet the education goal.
`deposit_frequency_interval`	The period interval associated with `deposit_amount` .
`projected_accumulation_savings`	The projected balance at the end of `accumulation_horizon` .
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`total_cost`	The total amount of withdrawals taken over `decumulation_horizon` .
`total_taxes`	The total taxes paid on withdrawals over `decumulation_horizon` .
`return_details`	Portfolio return information over the length of the horizon, broken down by period.

NAV

Field	Description
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`period_contribution`	The deposit made for this period.
`period_withdrawal`	The withdrawal made for this period.
`period_taxes`	The taxes paid during this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`cumulative_taxes`	The cumulative taxes paid up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

Percent of Costs Covered

A calculator to help plan for an education goal by solving for the percentage of the total cost that the user can afford.

NAV

HTTP REQUEST

```
POST /education_calculator/percent_covered
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
         "initial_balance": 5000,
         "accumulation_horizon": 3,
         "decumulation_horizon": 4,
         "total_annual_cost": 40000,
         "portfolio_return": 0.06,
         "education_inflation_rate": 0.05,
         "general_inflation_rate": 0.02,
         "tax_rate": 0.33,
         "deposit_schedule": {
            "deposit_amount": 4000,
            "deposit_frequency_interval": "year",
            "adjust_deposit_for_inflation": true
         }
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/education_calculator/p
```

ARGUMENTS

Parameter	Description
`accumulation_horizon` *integer, required*	The amount of years until funds are needed to pay for an education expense.
`decumulation_horizon` *integer, required*	The number of years to pay the `total_annual_cost`.
`total_annual_cost` *float, required*	The total cost paid per year for the education goal, represented in today's dollars.
`portfolio_return` *float, required*	The portfolio return for the length of the `accumulation_horizon` and `decumulation_horizon`.
`initial_balance` *float*	The amount currently saved for the education goal. If excluded, defaults to `0`.

NAV

Parameter	Description
`education_inflation_rate` *float*	The inflation rate attributed to `total_annual_cost`. If excluded, defaults to `0.05`.
`general_inflation_rate` *float*	The rate used to increase deposits for inflation. If excluded, defaults to `0.02`.
`tax_rate` *float*	The tax rate to apply to investments upon removal from the portfolio. If excluded, defaults to `0`.
`deposit_schedule` *map*	The deposit schedule for the education goal.
`deposit_amount` *float*	The amount deposited per a given period & number of intervals. Deposits are assumed to continue through the length of `accumulation_horizon` and `decumulation_horizon`. If excluded, defaults to `0`.
`deposit_frequency_interval` *string*	The period interval to be used in relation to the `deposit_amount`. The value may be one of the following: `year`, `quarter`, `month`, or `week`. If excluded, defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, the deposits should be increased over time with the rate of inflation. If excluded, defaults to `true`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

Example Response

```
{
    "achievable_cost": 5047.62,
```

NAV ☰


```
        "target_cost": 40000,
        "percent_of_costs_covered": 0.1262,
        "projected_accumulation_savings": 18935.88,
        "total_earnings": 4732.09,
        "total_contributions": 29737.13,
        "total_cost": 26444.38,
        "total_taxes": 13024.84,
        "return_details": {
            "0": {
                "period_earnings": 0,
                "period_contribution": 0,
                "period_withdrawal": 0,
                "period_taxes": 0,
                "cumulative_earnings": 0,
                "cumulative_contributions": 0,
                "cumulative_withdrawals": 0,
                "cumulative_taxes": 0,
                "ending_balance": 5000
            },
            "1": {
                "period_earnings": 300,
                "period_contribution": 4000,
                "period_withdrawal": 0,
                "period_taxes": 0,
                "cumulative_earnings": 300,
                "cumulative_contributions": 4000,
                "cumulative_withdrawals": 0,
                "cumulative_taxes": 0,
                "ending_balance": 9300
            },
            "2": {
                "period_earnings": 558,
                "period_contribution": 4080,
                "period_withdrawal": 0,
                "period_taxes": 0,
                "cumulative_earnings": 858,
                "cumulative_contributions": 8080,
                "cumulative_withdrawals": 0,
                "cumulative_taxes": 0,
                "ending_balance": 13938
            },
            "3": {
                "period_earnings": 836.28,
                "period_contribution": 4161.6,
                "period_withdrawal": 0,
                "period_taxes": 0,
```

NAV III


```
            "cumulative_earnings": 1694.28,
            "cumulative_contributions": 12241.6,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 18935.88
        },
        "4": {
            "period_earnings": 1136.15,
            "period_contribution": 4244.83,
            "period_withdrawal": -6135.41,
            "period_taxes": -3021.92,
            "cumulative_earnings": 2830.43,
            "cumulative_contributions": 16486.43,
            "cumulative_withdrawals": -6135.41,
            "cumulative_taxes": -3021.92,
            "ending_balance": 15159.54
        },
        "5": {
            "period_earnings": 909.57,
            "period_contribution": 4329.73,
            "period_withdrawal": -6442.18,
            "period_taxes": -3173.01,
            "cumulative_earnings": 3740.01,
            "cumulative_contributions": 20816.16,
            "cumulative_withdrawals": -12577.59,
            "cumulative_taxes": -6194.93,
            "ending_balance": 10783.65
        },
        "6": {
            "period_earnings": 647.02,
            "period_contribution": 4416.32,
            "period_withdrawal": -6764.29,
            "period_taxes": -3331.66,
            "cumulative_earnings": 4387.02,
            "cumulative_contributions": 25232.48,
            "cumulative_withdrawals": -19341.88,
            "cumulative_taxes": -9526.6,
            "ending_balance": 5751.04
        },
        "7": {
            "period_earnings": 345.06,
            "period_contribution": 4504.65,
            "period_withdrawal": -7102.5,
            "period_taxes": -3498.25,
            "cumulative_earnings": 4732.09,
            "cumulative_contributions": 29737.13,
```

NAV

```
                "cumulative_withdrawals": -26444.38,
                "cumulative_taxes": -13024.84,
                "ending_balance": 0
            }
        }
    }
```

RESPONSE

Field	Description
`achievable_cost`	The annual cost that can be covered, expressed in today's dollars.
`target_cost`	The `total_annual_cost` input representing the target annual goal amount.
`percent_of_costs_covered`	The percentage of `target_cost` that can be covered.
`projected_accumulation_savings`	The projected balance at the end of `accumulation_horizon`.
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`total_cost`	The total amount of withdrawals taken over `decumulation_horizon`.
`total_taxes`	The total taxes paid on withdrawals over `decumulation_horizon`.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`period_contribution`	The deposit made for this period.
`period_withdrawal`	The withdrawal made for this period.

NAV

Field	Description
`period_taxes`	The taxes paid during this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`cumulative_taxes`	The cumulative taxes paid up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

Total Annual Cost

A calculator to help plan for an education goal by solving for the total annual cost that a user could afford.

HTTP REQUEST

```
POST /education_calculator/annual_cost
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "initial_balance": 5000,
```

NAV

```
        "accumulation_horizon": 3,
        "decumulation_horizon": 4,
        "portfolio_return": 0.06,
        "percent_of_costs_covered": 1,
        "education_inflation_rate": 0.05,
        "general_inflation_rate": 0.02,
        "tax_rate": 0.33,
        "deposit_schedule": {
            "deposit_amount": 4000,
            "deposit_frequency_interval": "year",
            "adjust_deposit_for_inflation": true
        }
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/education_calculator/a
```

ARGUMENTS

Parameter	Description
`accumulation_horizon` *integer, required*	The amount of years until funds are needed to pay for an education expense.
`decumulation_horizon` *integer, required*	The number of years to pay the `total_annual_cost`.
`portfolio_return` *float, required*	The portfolio return for the length of the `accumulation_horizon` and `decumulation_horizon`.
`percent_of_costs_covered` *float*	The percentage of `total_annual_cost` to be paid for. If excluded, defaults to `1`.
`initial_balance` *float*	The amount currently saved for the education goal. If excluded, defaults to `0`.
`education_inflation_rate` *float*	The inflation rate attributed to `total_annual_cost`. If excluded, defaults to `0.05`.
`general_inflation_rate` *float*	The rate used to increase deposits for inflation. If excluded, defaults to `0.02`.
`tax_rate` *float*	The tax rate to apply to investments upon removal from the portfolio. If excluded, defaults to `0`.

NAV

Parameter	Description
`deposit_schedule` *map*	The deposit schedule for the education goal.
`deposit_amount` *float*	The amount deposited per a given period & number of intervals. Deposits are assumed to continue through the length of `accumulation_horizon` and `decumulation_horizon`. If excluded, defaults to `0`.
`deposit_frequency_interval` *string*	The period interval to be used in relation to the `deposit_amount`. The value may be one of the following: `year`, `quarter`, `month`, or `week`. If excluded, defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, the deposits should be increased over time with the rate of inflation. If excluded, defaults to `true`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

Example Response


```
{
    "total_annual_cost": 6135.41,
    "total_annual_cost_adjusted": 5047.62,
    "projected_accumulation_savings": 18935.88,
    "total_earnings": 4732.09,
    "total_contributions": 29737.13,
    "total_cost": 26444.38,
    "total_taxes": 13024.84,
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_contribution": 0,
```

NAV III


```
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 0,
            "cumulative_contributions": 0,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 5000
        },
        "1": {
            "period_earnings": 300,
            "period_contribution": 4000,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 300,
            "cumulative_contributions": 4000,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 9300
        },
        "2": {
            "period_earnings": 558,
            "period_contribution": 4080,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 858,
            "cumulative_contributions": 8080,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 13938
        },
        "3": {
            "period_earnings": 836.28,
            "period_contribution": 4161.6,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 1694.28,
            "cumulative_contributions": 12241.6,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 18935.88
        },
        "4": {
            "period_earnings": 1136.15,
            "period_contribution": 4244.83,
            "period_withdrawal": -6135.41,
            "period_taxes": -3021.92,
```

NAV


```
                "cumulative_earnings": 2830.43,
                "cumulative_contributions": 16486.43,
                "cumulative_withdrawals": -6135.41,
                "cumulative_taxes": -3021.92,
                "ending_balance": 15159.54
            },
            "5": {
                "period_earnings": 909.57,
                "period_contribution": 4329.73,
                "period_withdrawal": -6442.18,
                "period_taxes": -3173.01,
                "cumulative_earnings": 3740.01,
                "cumulative_contributions": 20816.16,
                "cumulative_withdrawals": -12577.59,
                "cumulative_taxes": -6194.93,
                "ending_balance": 10783.65
            },
            "6": {
                "period_earnings": 647.02,
                "period_contribution": 4416.32,
                "period_withdrawal": -6764.29,
                "period_taxes": -3331.66,
                "cumulative_earnings": 4387.02,
                "cumulative_contributions": 25232.48,
                "cumulative_withdrawals": -19341.88,
                "cumulative_taxes": -9526.6,
                "ending_balance": 5751.04
            },
            "7": {
                "period_earnings": 345.06,
                "period_contribution": 4504.65,
                "period_withdrawal": -7102.5,
                "period_taxes": -3498.25,
                "cumulative_earnings": 4732.09,
                "cumulative_contributions": 29737.13,
                "cumulative_withdrawals": -26444.38,
                "cumulative_taxes": -13024.84,
                "ending_balance": 0
            }
        }
    }
```


RESPONSE

NAV

Field	Description
`total_annual_cost`	The total education cost per year that can be afforded.
`total_annual_cost_adjusted`	The total education cost per year that can be afforded, represented in today's dollars.
`projected_accumulation_savings`	The projected balance at the end of `accumulation_horizon` .
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`total_cost`	The total amount of withdrawals taken over `decumulation_horizon` .
`total_taxes`	The total taxes paid on withdrawals over `decumulation_horizon` .
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`period_contribution`	The deposit made for this period.
`period_withdrawal`	The withdrawal made for this period.
`period_taxes`	The taxes paid during this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.

NAV

Field	Description
`cumulative_taxes`	The cumulative taxes paid up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

Emergency Fund Calculator

An emergency fund refers to a pool of money a user can rely on if they experience a loss of income. This fund should be able to cover all necessary expenses that the user would have to continue paying. The tool allows for the flexibility to use pre-set expense fields, or utilize an area for custom expenses. After specifying the number of months of expenses the emergency fund should cover, the calculator returns a total recommendation along with a plan to save for the total recommended amount in a variety of horizon intervals.

HTTP REQUEST

`POST /emergency_fund_calculator`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "emergency_fund_duration": 12,
        "housing_cost": 1650,
        "utility_payments": 60,
        "telecom_payments": 50,
        "insurance_payments": 0,
```

NAV

```
        "debt_payments": 0,
        "transportation_costs": 150,
        "food_costs": 1500,
        "other_expenses":
        {
            "entertainment": 50,
            "daycare": 150
        },
        "current_emergency_fund_balance": 15000,
        "interest_rate": 0.015,
        "savings_horizon": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
        "frequency_unit": "month"
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/emergency_fund_calculatc
```

ARGUMENTS

Parameter	Description
`emergency_fund_duration` *integer, required*	The number of periods (defined in `frequency_unit`) that the emergency fund will last.
`housing_cost` *float*	The user's housing costs, such as payments for rent, mortgage, maintenance, etc. If excluded, defaults to `0`.
`utility_payments` *float*	The user's utility payments, such as electricity, water, gas, etc. If excluded, defaults to `0`.
`telecom_payments` *float*	The user's telecom payments, such as internet, cell phone, cable, etc. If excluded, defaults to `0`.
`insurance_payments` *float*	The user's payments for insurance, such as auto, home, motorcycle, etc. If excluded, defaults to `0`.
`debt_payments` *float*	The user's debt payments, such as credit cards and loans. If excluded, defaults to `0`.
`transportation_costs` *float*	The user's transportation costs, such as gasoline, car payments, etc. If excluded, defaults to `0`.
`food_costs` *float*	The user's food costs, such as groceries, restaurants, etc. If excluded, defaults to `0`.

NAV

Parameter	Description
`other_expenses` *map*	A field to add any other expense type to be included in the emergency fund. If excluded, no additional expenses are included in the calculation.
`category_name` *string, required*	The category name for each additional expense.
`amount` *float, required*	The user's total expense for each given `category_name`.
`current_emergency_fund_balance` *float*	The user's current balance of their emergency fund. If excluded, defaults to `0`.
`interest_rate` *float*	The annualized interest rate earned on the `current_emergency_fund_balance` used to calculate the recommended savings amount. The `interest_rate` is converted from annual rate to the appropriate frequency defined in `frequency_unit`. If excluded, defaults to `0`.
`savings_horizon` *array*	The periods in the savings horizon to be used in the recommended savings schedule. The period frequency is defined by `frequency_unit`. If excluded, defaults to `[3, 6, 9, 12, 15, 18, 21, 24]`.
`frequency_unit` *string*	The frequency unit that applies to `emergency_fund_duration`, all expenses, `savings_horizon` and `saving_schedule`. The value may be one of the following: `year`, `quarter`, `month`, `two_weeks` or `week`. If excluded, defaults to `month`.
`client_id` *UUID*	The ID of a Nucleus client used to derive one or more of the following based on the client's spending history: `housing_cost`, `utility_payments`, `telecom_payments`, `insurance_payments`, `debt_payments`, `transportation_costs`, and `food_costs`. If raw values for any of these optional parameters are not provided, we will use `client_id` to try to derive those values instead.
`lookback_periods` *integer*	Number of periods to analyze when deriving expenses using a `client_id`. The unit of time associated with each period is based on `frequency_unit`, and expenses are averaged across all periods. Defaults to 3.

NAV

Parameter	Description
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `current_emergency_fund_balance`. If `current_emergency_fund_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive those values instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `current_emergency_fund_balance`. If `current_emergency_fund_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive those values instead.

Example Response


```
{
    "emergency_fund_recommendation": 43320,
    "savings_schedule": {
        "1": 28320.0,
        "2": 14151.22,
        "3": 9428.29,
        "4": 7066.83,
        "5": 5649.95,
        "6": 4705.37,
        "7": 4030.67,
        "8": 3524.65,
        "9": 3131.07,
        "10": 2816.21,
        "11": 2558.6,
        "12": 2343.93,
        "13": 2162.28,
        "14": 2006.58,
        "15": 1871.65,
        "16": 1753.58,
        "17": 1649.4,
        "18": 1556.8,
        "19": 1473.94,
        "20": 1399.37,
        "21": 1331.91,
        "22": 1270.57,
        "23": 1214.57,
```

NAV

```
            "24": 1163.24
        }
    }
```

RESPONSE

Field	Description
`emergency_fund_recommendation`	The total amount recommended for the user's emergency fund.
`savings_schedule`	The amount the user would need to save in order to achieve their emergency fund goal in a predefined amount of time.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`
- `Aggregation Account Transaction`

Only one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, and `portfolio_id` should be provided. An error will be thrown if multiple Nucleus data entities are referenced in a single request.

Life Insurance Needs Calculator

When deciding to purchase life insurance, users face a common question: how much life insurance do I need? While some in the industry rely on broad rules of thumb (such as a multiple of household income), this calculator uses a discounted cash flow approach. Once the user provides expense and other pertinent information, the calculator returns the life insurance need, a breakdown of expenses by category, and return details broken down by period.

HTTP REQUEST

NAV

POST /life_insurance/needs_calculator

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "mortgage_balance": 200000,
        "other_debt": 100000,
        "interest_rate": 0.06,
        "end_of_life_expenses": 10000,
        "existing_life_insurance": 50000,
        "liquid_assets": 400000,
        "general_inflation_rate": 0.02,
        "education_inflation_rate": 0.05,
        "tax_rate": 0.25,
        "benefit_amount_rounding": 6,
        "margin_of_error": 0.05,
        "children_education_config":
        [
          {
            "current_age": 17,
            "education_config":
            [
              {
                "start_age": 18,
                "end_age": 22,
                "total_annual_cost": 40000
              }
            ]
          }
        ],
        "income_config":
        [
          {
            "annual_net_take_home_pay": 50000,
            "percentage_of_income_covered": 1,
            "income_benefit_duration": 5
          }
        ],
        "beneficiary_bequest_config":
        [
          {
            "years_until_bequest": 1,
```

NAV

```
            "bequest_amount": 10000,
            "bequest_duration": 5
          }
        ]
}' "https://[sandbox][api].hydrogenplatform.com/proton/v1/life_insurance/needs_calcu
```

ARGUMENTS

Parameter	Description
`mortgage_balance` *float, conditioanl requirement*	The user's outstanding mortgage balance.
`other_debt` *integer, conditional requirement*	Other outstanding debt.
`interest_rate` *float, required*	The annual interest rate earned once the benefit amount is received.
`end_of_life_expenses` *float*	Burial, funeral, and other end-of-life expenses to be paid at the first period. If excluded, defaults to `0`.
`existing_life_insurance` *float*	Existing life insurance currently held by the insuree. If excluded, defaults to `0`.
`liquid_assets` *float*	Current liquid assets owned by the insuree and beneficiary. If excluded, defaults to `0`.
`general_inflation_rate` *float*	The inflation rate to be applied to the `benefit_amount` and `annual_net_take_home_pay`. If excluded, defaults to `0`.
`education_inflation_rate` *float*	The inflation rate to be applied to all tuition. If excluded, defaults to `0`.
`tax_rate` *float*	The tax rate to be applied to investment earnings from the `interest_rate`. Life insurance benefits are assumed to be tax-free. If excluded, defaults to `0`.
`benefit_amount_rounding` *integer*	A parameter to round the benefit amount up to a configurable number of digits. This parameter should be utilized if a firm will not underwrite exact life insurance amounts, but will round to the nearest amount. `0`: round to the hundredth, `1`: round to the

NAV

Parameter	Description
	tenth, `2` : round to a single digit, `3` : round to two digits, etc. If excluded, defaults to `0` .
`margin_of_error` *float*	The margin of error to apply to the life insurance needed, before rounding. A `margin_of_error` of `0.10` would return 110% of the total insurance needed. If excluded, defaults to `0` .
`children_education_config` *map*	Benefit information to provide for a child's primary, secondary and college education.
`current_age` *integer, required*	The child's age as of the first period.
`education_config` *map, required*	A configuration of the child's education cost.
`start_age` *integer, required*	The child's age at the beginning of the first education year.
`end_age` *integer, required*	The child's age at the beginning of the last education year.
`total_annual_cost` *float, required*	The total annual cost of the child's education.
`income_config` *map*	Information on replacement income for beneficiaries. Income is assumed to increase with the rate of inflation.
`annual_net_take_home_pay` *float, required*	The policy owner's annual after-tax income.
`percentage_of_income_covered` *float*	The percentage of `annual_net_take_home_pay` needed by beneficiaries. If excluded, defaults to `1` .
`income_benefit_duration` *integer*	The number of years needed to provide replacement income to beneficiaries, starting at the first period. If excluded, defaults to `10` .
`beneficiary_bequest_config` *map*	Information on additional benefits needed by beneficiaries.
`bequest_amount` *float, required*	The amount of the benefit, paid annually.

NAV

Parameter	Description
`years_until_bequest` *integer*	The number of years until the benefit is needed. A value of `0` indicates the benefit would be needed immediately (the first period). If excluded, defaults to `0`.
`bequest_duration` *integer*	The amount of years to repeat the `benefit_amount`. If excluded, defaults to `10`.
`client_id` *UUID, conditional requirement*	The ID of a Nucleus client used to derive one or more of the following based on the client's financials: `mortgage_balance`, `other_debt`, and `liquid_assets`. If required parameters `mortgage_balance` and/or `other_debt` are not provided, `client_id` is required. If a raw value for optional parameter `liquid_assets` is not provided, we will use `client_id` to try to derive this value instead.

Example Response


```
{
    "life_insurance_needed": 390000,
    "total_life_insurance_needed": 440000,
    "life_insurance_needs_breakdown": {
        "mortgage": 200000,
        "other": 100000,
        "education": 213033.65,
        "income_replacement": 243087.46,
        "beneficiary_bequest": 47454.39,
        "end_of_life": 10000
    },
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_withdrawal": 0,
            "cumulative_earnings": 0,
            "cumulative_withdrawals": 0,
            "ending_balance": 840000
        },
        "1": {
            "period_earnings": 0,
            "period_withdrawal": 361000,
            "cumulative_earnings": 0,
```

NAV


```
                "cumulative_withdrawals": 361000,
                "ending_balance": 479000
            },
            "2": {
                "period_earnings": 21555,
                "period_withdrawal": 106524,
                "cumulative_earnings": 21555,
                "cumulative_withdrawals": 467524,
                "ending_balance": 394031
            },
            "3": {
                "period_earnings": 17731.39,
                "period_withdrawal": 109977.48,
                "cumulative_earnings": 39286.39,
                "cumulative_withdrawals": 577501.48,
                "ending_balance": 301784.91
            },
            "4": {
                "period_earnings": 13580.32,
                "period_withdrawal": 113566.18,
                "cumulative_earnings": 52866.72,
                "cumulative_withdrawals": 691067.66,
                "ending_balance": 201799.06
            },
            "5": {
                "period_earnings": 9080.96,
                "period_withdrawal": 117296.11,
                "cumulative_earnings": 61947.67,
                "cumulative_withdrawals": 808363.77,
                "ending_balance": 93583.9
            },
            "6": {
                "period_earnings": 4211.28,
                "period_withdrawal": 64865.45,
                "cumulative_earnings": 66158.95,
                "cumulative_withdrawals": 873229.22,
                "ending_balance": 32929.73
            }
        }
    }
```


RESPONSE

NAV

Field	Description
`life_insurance_needed`	The supplemental life insurance needed by the user.
`total_life_insurance_needed`	The total life insurance needed by the user, including the current life insurance.
`life_insurance_needs_breakdown`	A breakdown of what the life insurance proceeds will be used for.
`mortgage`	Insurance required to cover existing mortgage balance, paid immediately.
`other`	Insurance required to cover other debt.
`education`	Insurance required to cover future education expenses.
`income_replacement`	Insurance required to cover loss of income.
`beneficiary_bequest`	Insurance required to cover amount being bequested.
`end_of_life`	Insurance required to cover end of life expenses.
`return_details`	Portfolio return information over the length of the horizon, broken down by period. Period `0` represents the assets available including the calculated insurance payment. Period `1` reflects any immediate expenses, and the remaining balance is drawn down starting with period `2` to cover ongoing expenses.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`period_withdrawal`	The withdrawal made for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.

NAV

Field	Description
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Aggregation Account`
- `Aggregation Account Balance`
- `Account Categories`

Major Purchase Calculator

When planning to make a major purchase, such as a down payment on a home/car, investors need to determine how much is needed and how to stay on track. After collecting information about the user's major purchase, this tool solves for three different parameters: deposit amount, purchase amount, and purchase horizon.

Deposit Amount

A calculator to help determine the deposit amount needed to meet the major purchase goal.

HTTP REQUEST

`POST /purchase_calculator/deposit_amount`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "purchase_horizon": 5,
        "purchase_amount": 50000,
        "portfolio_return": 0.06,
        "horizon_frequency_interval": "year",
```

NAV

```
        "current_savings": 0,
        "deposit_schedule": {
            "deposit_frequency_interval": "year",
            "adjust_deposit_for_inflation": true
        },
        "inflation_rate": 0.02,
        "investment_tax": 0.25
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/purchase_calculator/depc
```

ARGUMENTS

Parameter	Description
`purchase_horizon` *integer, required*	The amount of time until the major purchase is made.
`purchase_amount` *float, required*	The amount of the major purchase.
`portfolio_return` *float, required*	The portfolio's return, calculated on an annual basis.
`horizon_frequency_interval` *string*	The interval at which to measure the purchase horizon. The value may be one of the following: `year`, `quarter`, `month`, or `week`. Earnings are calculated at the same frequency as `horizon_frequency_interval`. If excluded, defaults to `year`.
`current_savings` *float*	The current amount of savings earmarked for the goal. If excluded, defaults to `0`.
`deposit_schedule` *map*	The deposit schedule for the major purchase goal.
`deposit_frequency_interval` *string*	The period interval to be used in relation to the `deposit_amount`. The value may be one of the following: `year`, `quarter`, `month`, or `week`. If excluded, defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, the deposits are increased over time with the rate of inflation. If excluded, defaults to `true`.
`inflation_rate`	The annual inflation rate of the `purchase_amount`. If excluded, defaults to `0`.

NAV

Parameter	Description
`investment_tax` *float*	The tax on the investments used to pay for the major purchase. If excluded, defaults to `0`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `current_savings`. If `current_savings` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `current_savings`. If `current_savings` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

Example Response


```
{
    "deposit_amount": 12574.34,
    "deposit_frequency_interval": "year",
    "adjusted_goal_amount": 73605.39,
    "projected_savings_at_purchase_horizon": 73605.39,
    "total_earnings": 8168.03,
    "total_contributions": 65437.36,
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_contribution": 0,
            "cumulative_earnings": 0,
            "cumulative_contributions": 0,
            "ending_balance": 0
        },
        "1": {
            "period_earnings": 0,
            "period_contribution": 12574.34,
            "cumulative_earnings": 0,
            "cumulative_contributions": 12574.34,
            "ending_balance": 12574.34
        },
        "2": {
            "period_earnings": 754.46,
            "period_contribution": 12825.82,
```

NAV


```
                "cumulative_earnings": 754.46,
                "cumulative_contributions": 25400.16,
                "ending_balance": 26154.62
            },
            "3": {
                "period_earnings": 1569.28,
                "period_contribution": 13082.34,
                "cumulative_earnings": 2323.74,
                "cumulative_contributions": 38482.5,
                "ending_balance": 40806.24
            },
            "4": {
                "period_earnings": 2448.37,
                "period_contribution": 13343.99,
                "cumulative_earnings": 4772.11,
                "cumulative_contributions": 51826.49,
                "ending_balance": 56598.6
            },
            "5": {
                "period_earnings": 3395.92,
                "period_contribution": 13610.87,
                "cumulative_earnings": 8168.03,
                "cumulative_contributions": 65437.36,
                "ending_balance": 73605.39
            }
        }
    }
```


RESPONSE

Field	Description
`deposit_amount`	The amount to deposit in order to meet the purchase goal.
`deposit_frequency_interval`	The frequency interval of the deposit.
`adjusted_goal_amount`	The effective goal target amount, adjusted for taxes and inflation.
`projected_savings_at_purchase_horizon`	The total amount of savings that are projected to be available at the final horizon, expressed in today's dollars.

NAV

Field	Description
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions are made.
`period_contribution`	The deposit made for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

Purchase Amount

A calculator to help determine the amount that the user could afford for the major purchase, given the other inputs.

HTTP REQUEST

```
POST /purchase_calculator/amount
```

NAV

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "purchase_horizon": 6,
        "portfolio_return": 0.06,
        "horizon_frequency_interval": "year",
        "current_savings": 0,
        "deposit_schedule": {
            "deposit_amount": 10000,
            "deposit_frequency_interval": "year",
            "adjust_deposit_for_inflation": true
        },
        "inflation_rate": 0.01,
        "investment_tax": 0.2
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/purchase_calculator/amou
```

ARGUMENTS

Parameter	Description
`purchase_horizon` *integer, required*	The amount of time until the major purchase is made.
`portfolio_return` *float, required*	The portfolio's return, calculated on an annual basis.
`horizon_frequency_interval` *string*	The interval at which to measure the `purchase_horizon`. The value may be one of the following: `year`, `quarter`, `month`, or `week`. Earnings are calculated at the same frequency as `horizon_frequency_interval`. If excluded, defaults to `year`.
`current_savings` *float*	The current amount of savings earmarked for the goal. If excluded, defaults to `0`.
`deposit_schedule` *map*	The deposit schedule for the major purchase goal.
`deposit_amount` *float*	The periodic additions to the major purchase goal. If excluded, defaults to `0`.

NAV

Parameter	Description
`deposit_frequency_interval` *string*	The period interval to be used in relation to the `deposit_amount`. The value may be one of the following: `year`, `quarter`, `month`, or `week`. If excluded, defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, the deposits are increased over time with the rate of inflation. If excluded, defaults to `true`.
`inflation_rate` *float*	The annual inflation rate of the `purchase_amount`. If excluded, defaults to `0`.
`investment_tax` *float*	The tax on the investments used to pay for the major purchase. If excluded, defaults to `0`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `current_savings`. If `current_savings` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `current_savings`. If `current_savings` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

Example Response


```
{
    "purchase_amount": 57119.83,
    "purchase_amount_adjusted": 53809.47,
    "projected_savings_at_purchase_horizon": 71399.79,
    "total_earnings": 9879.64,
    "total_contributions": 61520.15,
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_contribution": 0,
            "cumulative_earnings": 0,
            "cumulative_contributions": 0,
            "ending_balance": 0
        },
```

NAV


```
            "1": {
                "period_earnings": 0,
                "period_contribution": 10000,
                "cumulative_earnings": 0,
                "cumulative_contributions": 10000,
                "ending_balance": 10000
            },
            "2": {
                "period_earnings": 600,
                "period_contribution": 10100,
                "cumulative_earnings": 600,
                "cumulative_contributions": 20100,
                "ending_balance": 20700
            },
            "3": {
                "period_earnings": 1242,
                "period_contribution": 10201,
                "cumulative_earnings": 1842,
                "cumulative_contributions": 30301,
                "ending_balance": 32143
            },
            "4": {
                "period_earnings": 1928.58,
                "period_contribution": 10303.01,
                "cumulative_earnings": 3770.58,
                "cumulative_contributions": 40604.01,
                "ending_balance": 44374.59
            },
            "5": {
                "period_earnings": 2662.48,
                "period_contribution": 10406.04,
                "cumulative_earnings": 6433.06,
                "cumulative_contributions": 51010.05,
                "ending_balance": 57443.11
            },
            "6": {
                "period_earnings": 3446.59,
                "period_contribution": 10510.1,
                "cumulative_earnings": 9879.64,
                "cumulative_contributions": 61520.15,
                "ending_balance": 71399.79
            }
        }
    }
```

NAV

RESPONSE

Field	Description
`purchase_amount`	The amount of the major purchase.
`purchase_amount_adjusted`	The amount of the major purchase, represented in today's dollars.
`projected_savings_at_purchase horizon`	The total amount of savings that are projected to be available at `purchase_horizon`, expressed in today's dollars.
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The earnings for this period. Earnings are calculated at the beginning of each period, before contributions are made.
`period_contribution`	The deposit made for this period.
`cumulative_earnings`	The cumulative earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

Purchase Horizon

A calculator to help determine the amount of years needed to save for a major purchase.

NAV

HTTP REQUEST

```
POST /purchase_calculator/horizon
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "purchase_amount": 50000,
        "portfolio_return": 0.06,
        "current_savings": 0,
        "deposit_schedule": {
            "deposit_amount": 10000,
            "deposit_frequency_interval": "year",
            "adjust_deposit_for_inflation": true
        },
        "inflation_rate": 0.02,
        "investment_tax": 0.25
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/purchase_calculator/hori
```

ARGUMENTS

Parameter	Description
`purchase_amount` *float, required*	The amount of the major purchase.
`portfolio_return` *float, required*	The portfolio's return, calculated on an annual basis.
`current_savings` *float*	The current amount of savings earmarked for the goal. If excluded, defaults to `0`.
`deposit_schedule` *map*	The deposit schedule for the major purchase goal.
`deposit_amount` *float*	The periodic additions to the major purchase goal. If excluded, defaults to `0`.
`deposit_frequency_interval` *string*	The period interval to be used in relation to the `deposit_amount`. The value may be one of the following:

NAV

Parameter	Description
	`year`, `quarter`, `month`, or `week`. If excluded, defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, the deposits are increased over time with the rate of inflation. If excluded, defaults to `true`.
`inflation_rate` *float*	The annual inflation rate of the `purchase_amount`. If excluded, defaults to `0`.
`investment_tax` *float*	The tax on the investments used to pay for the major purchase. If excluded, defaults to `0`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `current_savings`. If `current_savings` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `current_savings`. If `current_savings` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

Example Response


```
{
    "purchase_horizon": 7,
    "purchase_horizon_frequency_interval": "year",
    "adjusted_goal_amount": 76579.04,
    "projected_savings_at_purchase_horizon": 88736.15,
    "total_earnings": 14393.31,
    "total_contributions": 74342.83,
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_contribution": 0,
            "cumulative_earnings": 0,
            "cumulative_contributions": 0,
            "ending_balance": 0
        },
        "1": {
```

NAV III


```
            "period_earnings": 0,
            "period_contribution": 10000,
            "cumulative_earnings": 0,
            "cumulative_contributions": 10000,
            "ending_balance": 10000
        },
        "2": {
            "period_earnings": 600,
            "period_contribution": 10200,
            "cumulative_earnings": 600,
            "cumulative_contributions": 20200,
            "ending_balance": 20800
        },
        "3": {
            "period_earnings": 1248,
            "period_contribution": 10404,
            "cumulative_earnings": 1848,
            "cumulative_contributions": 30604,
            "ending_balance": 32452
        },
        "4": {
            "period_earnings": 1947.12,
            "period_contribution": 10612.08,
            "cumulative_earnings": 3795.12,
            "cumulative_contributions": 41216.08,
            "ending_balance": 45011.2
        },
        "5": {
            "period_earnings": 2700.67,
            "period_contribution": 10824.32,
            "cumulative_earnings": 6495.79,
            "cumulative_contributions": 52040.4,
            "ending_balance": 58536.19
        },
        "6": {
            "period_earnings": 3512.17,
            "period_contribution": 11040.81,
            "cumulative_earnings": 10007.96,
            "cumulative_contributions": 63081.21,
            "ending_balance": 73089.17
        },
        "7": {
            "period_earnings": 4385.35,
            "period_contribution": 11261.62,
            "cumulative_earnings": 14393.31,
            "cumulative_contributions": 74342.83,
```

NAV

```
                "ending_balance": 88736.15
            }
        }
    }
```

RESPONSE

Field	Description
`purchase_horizon`	The number of periods needed in order to meet the major purchase goal.
`purchase_horizon_frequency_interval`	The unit of time associated with `purchase_horizon`. Will always be `year`.
`adjusted_goal_amount`	The effective goal target amount, adjusted for taxes and inflation.
`projected_savings_at_purchase_horizon`	The total amount of savings that are projected to be available at the major purchase date, expressed in today's dollars.
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The earnings for this period. Earnings are calculated at the beginning of each period, before contributions are made.
`period_contribution`	The deposit made for this period.
`cumulative_earnings`	The cumulative earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.

NAV

Field	Description
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

Mortgage Calculator

Many users face the question of how much a mortgage may cost when deciding to buy a property. This tool simplifies the parameters to five components: the mortgage interest rate, mortgage term, home price, down payment, and the periodic (monthly) payment. Of those five parameters, this mortgage calculator provides three different endpoints to solve for the following parameters: a periodic (monthly) payment, down payment and total home price.

Down Payment

This calculator helps a user solve for the down payment they will need to pay, given the other factors in their home purchase.

HTTP REQUEST

`POST /mortgage_calculator/down_payment`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "home_price": 100000,
        "periodic_payment": 8000,
```

NAV

```
        "interest_rate": 0.04,
        "mortgage_term": 6
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/mortgage_calculator/down
```

ARGUMENTS

Parameter	Description
`home_price` *float, required*	The total cost of the home, including down payments but excluding transactional costs of purchasing the home (e.g. insurance, legal fees, closing costs, etc.)
`periodic_payment` *float, required*	The periodic monthly payment for the mortgage.
`interest_rate` *float*	The annualized interest rate charged on the balance of the mortgage. If excluded, defaults to `0.0`.
`mortgage_term` *integer*	The length of time the mortgage will take place, as expressed in months. If excluded, defaults to `360`.

Example Response


```
{
    "down_payment": 52545.22,
    "total_payment": 48000,
    "total_principal": 47454.78,
    "total_interest": 545.22,
    "total_home_cost": 100545.22,
    "schedule": {
        "0": {
            "Payment": 0,
            "Principal": 0,
            "Interest": 0,
            "Cumulative_Payment": 0,
            "Cumulative_Principal": 0,
            "Cumulative_Interest": 0,
            "Balance": 47454.78
        },
        "1": {
            "Payment": 8000,
            "Principal": 7844.65,
```

NAV III


```
            "Interest": 155.35,
            "Cumulative_Payment": 8000,
            "Cumulative_Principal": 7844.65,
            "Cumulative_Interest": 155.35,
            "Balance": 39610.13
        },
        "2": {
            "Payment": 8000,
            "Principal": 7870.33,
            "Interest": 129.67,
            "Cumulative_Payment": 16000,
            "Cumulative_Principal": 15714.97,
            "Cumulative_Interest": 285.03,
            "Balance": 31739.81
        },
        "3": {
            "Payment": 8000,
            "Principal": 7896.09,
            "Interest": 103.91,
            "Cumulative_Payment": 24000,
            "Cumulative_Principal": 23611.06,
            "Cumulative_Interest": 388.94,
            "Balance": 23843.71
        },
        "4": {
            "Payment": 8000,
            "Principal": 7921.94,
            "Interest": 78.06,
            "Cumulative_Payment": 32000,
            "Cumulative_Principal": 31533.01,
            "Cumulative_Interest": 466.99,
            "Balance": 15921.77
        },
        "5": {
            "Payment": 8000,
            "Principal": 7947.88,
            "Interest": 52.12,
            "Cumulative_Payment": 40000,
            "Cumulative_Principal": 39480.88,
            "Cumulative_Interest": 519.12,
            "Balance": 7973.9
        },
        "6": {
            "Payment": 8000,
            "Principal": 7973.9,
            "Interest": 26.1,
```

NAV

```
                "Cumulative_Payment": 48000,
                "Cumulative_Principal": 47454.78,
                "Cumulative_Interest": 545.22,
                "Balance": 0
            }
        }
    }
```

RESPONSE

Field	Description
`down_payment`	The payment due upfront when buying a home, given the other inputs provided by the user.
`total_payment`	The total mortgage payments made over `mortgage_term`.
`total_principal`	The total amount that went toward the mortgage principal.
`total_interest`	The total amount that went toward the mortgage interest.
`total_home_cost`	The total cost of the home, including down payment and all mortgage payments. This value is greater than `home_price` when `interest_rate` is greater than 0.
`schedule`	Details for the mortgage payment, broken down by period.
`Payment`	The total payment made for this period, consisting of interest and principal.
`Principal`	The principal payment made for this period.
`Interest`	The interest payment made for this period.
`Cumulative_Payment`	The cumulative total payment made up to and including this period.
`Cumulative_Principal`	The cumulative principal payment made up to and including this period.

NAV

Field	Description
`Cumulative_Interest`	The cumulative interest payment made up to and including this period.
`Balance`	The remaining mortgage balance at the end of the period.

Home Price

This calculator helps a user solve for the total home price they can afford, given the other factors related to their home purchase.

HTTP REQUEST

`POST /mortgage_calculator/home_price`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "down_payment": 50000,
        "periodic_payment": 8000,
        "interest_rate": 0.04,
        "mortgage_term": 6
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/mortgage_calculator/home
```

ARGUMENTS

Parameter	Description
`down_payment` *float, required*	The payment due upfront when buying a home.
`periodic_payment` *float, required*	The periodic monthly payment for the mortgage.
`interest_rate`	The annualized interest rate charged on the balance of the mortgage. If excluded, defaults to `0.0`.

NAV

Parameter	Description
`mortgage_term` *integer*	The length of time the mortgage will take place, as expressed in months. If excluded, defaults to `360`.

Example Response


```
{
    "home_price": 97454.78,
    "total_payment": 48000,
    "total_principal": 47454.78,
    "total_interest": 545.22,
    "total_home_cost": 98000,
    "schedule": {
        "0": {
            "Payment": 0,
            "Principal": 0,
            "Interest": 0,
            "Cumulative_Payment": 0,
            "Cumulative_Principal": 0,
            "Cumulative_Interest": 0,
            "Balance": 47454.78
        },
        "1": {
            "Payment": 8000,
            "Principal": 7844.65,
            "Interest": 155.35,
            "Cumulative_Payment": 8000,
            "Cumulative_Principal": 7844.65,
            "Cumulative_Interest": 155.35,
            "Balance": 39610.13
        },
        "2": {
            "Payment": 8000,
            "Principal": 7870.33,
            "Interest": 129.67,
            "Cumulative_Payment": 16000,
            "Cumulative_Principal": 15714.97,
            "Cumulative_Interest": 285.03,
            "Balance": 31739.81
        },
        "3": {
            "Payment": 8000,
```

NAV


```
                "Principal": 7896.09,
                "Interest": 103.91,
                "Cumulative_Payment": 24000,
                "Cumulative_Principal": 23611.06,
                "Cumulative_Interest": 388.94,
                "Balance": 23843.71
            },
            "4": {
                "Payment": 8000,
                "Principal": 7921.94,
                "Interest": 78.06,
                "Cumulative_Payment": 32000,
                "Cumulative_Principal": 31533.01,
                "Cumulative_Interest": 466.99,
                "Balance": 15921.77
            },
            "5": {
                "Payment": 8000,
                "Principal": 7947.88,
                "Interest": 52.12,
                "Cumulative_Payment": 40000,
                "Cumulative_Principal": 39480.88,
                "Cumulative_Interest": 519.12,
                "Balance": 7973.9
            },
            "6": {
                "Payment": 8000,
                "Principal": 7973.9,
                "Interest": 26.1,
                "Cumulative_Payment": 48000,
                "Cumulative_Principal": 47454.78,
                "Cumulative_Interest": 545.22,
                "Balance": 0
            }
        }
    }
```


RESPONSE

Field	Description
`home_price`	The total cost of the home that can be afforded, given the other inputs provided by the user.

NAV

Field	Description
`total_payment`	The total mortgage payments made over `mortgage_term`.
`total_principal`	The total amount that went toward the mortgage principal.
`total_interest`	The total amount that went toward the mortgage interest.
`total_home_cost`	The total cost of the home, including down payment and all mortgage payments. This value is greater than `home_price` when `interest_rate` is greater than 0.
`schedule`	Details for the mortgage payment, broken down by period.
`Payment`	The total payment made for this period, consisting of interest and principal.
`Principal`	The principal payment made for this period.
`Interest`	The interest payment made for this period.
`Cumulative_Payment`	The cumulative total payment made up to and including this period.
`Cumulative_Principal`	The cumulative principal payment made up to and including this period.
`Cumulative_Interest`	The cumulative interest payment made up to and including this period.
`Balance`	The remaining mortgage balance at the end of the period.

Periodic Payment

This calculator helps a user solve for the monthly mortgage payment, given the other factors in their home purchase.

HTTP REQUEST

```
POST /mortgage_calculator/periodic_payment
```

NAV

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "home_price": 100000,
        "down_payment": 50000,
        "interest_rate": 0.04,
        "mortgage_term": 6
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/mortgage_calculator/peri
```

ARGUMENTS

Parameter	Description
`home_price` *float, required*	The total cost of the home, including down payments but excluding transactional costs of purchasing the home (e.g. insurance, legal fees, closing costs, etc.)
`down_payment` *float, required*	The payment due upfront when buying a home.
`interest_rate` *float*	The annualized interest rate charged on the balance of the mortgage. If excluded, defaults to `0.0`.
`mortgage_term` *integer*	The length of time the mortgage will take place, as expressed in months. If excluded, defaults to `360`.

Example Response

```
{
    "periodic_payment": 8429.08,
    "total_payment": 50574.46,
    "total_principal": 50000,
    "total_interest": 574.46,
    "total_home_cost": 100574.46,
    "schedule": {
        "0": {
            "Payment": 0,
            "Principal": 0,
```

NAV


```
            "Interest": 0,
            "Cumulative_Payment": 0,
            "Cumulative_Principal": 0,
            "Cumulative_Interest": 0,
            "Balance": 50000
        },
        "1": {
            "Payment": 8429.08,
            "Principal": 8265.39,
            "Interest": 163.69,
            "Cumulative_Payment": 8429.08,
            "Cumulative_Principal": 8265.39,
            "Cumulative_Interest": 163.69,
            "Balance": 41734.61
        },
        "2": {
            "Payment": 8429.08,
            "Principal": 8292.45,
            "Interest": 136.63,
            "Cumulative_Payment": 16858.15,
            "Cumulative_Principal": 16557.84,
            "Cumulative_Interest": 300.32,
            "Balance": 33442.16
        },
        "3": {
            "Payment": 8429.08,
            "Principal": 8319.6,
            "Interest": 109.48,
            "Cumulative_Payment": 25287.23,
            "Cumulative_Principal": 24877.44,
            "Cumulative_Interest": 409.8,
            "Balance": 25122.56
        },
        "4": {
            "Payment": 8429.08,
            "Principal": 8346.83,
            "Interest": 82.24,
            "Cumulative_Payment": 33716.31,
            "Cumulative_Principal": 33224.27,
            "Cumulative_Interest": 492.04,
            "Balance": 16775.73
        },
        "5": {
            "Payment": 8429.08,
            "Principal": 8374.16,
            "Interest": 54.92,
```

NAV


```
            "Cumulative_Payment": 42145.39,
            "Cumulative_Principal": 41598.43,
            "Cumulative_Interest": 546.96,
            "Balance": 8401.57
        },
        "6": {
            "Payment": 8429.08,
            "Principal": 8401.57,
            "Interest": 27.5,
            "Cumulative_Payment": 50574.46,
            "Cumulative_Principal": 50000,
            "Cumulative_Interest": 574.46,
            "Balance": 0
        }
    }
}
```


RESPONSE

Field	Description
`periodic_payment`	The periodic monthly payment for the mortgage, given the other inputs provided by the user.
`total_payment`	The total mortgage payments made over `mortgage_term`.
`total_principal`	The total amount that went toward the mortgage principal.
`total_interest`	The total amount that went toward the mortgage interest.
`total_home_cost`	The total cost of the home, including down payment and all mortgage payments. This value is greater than `home_price` when `interest_rate` is greater than 0.
`schedule`	Details for the mortgage payment, broken down by period.
`Payment`	The total payment made for this period, consisting of interest and principal.
`Principal`	The principal payment made for this period.

NAV

Field	Description
`Interest`	The interest payment made for this period.
`Cumulative_Payment`	The cumulative total payment made up to and including this period.
`Cumulative_Principal`	The cumulative principal payment made up to and including this period.
`Cumulative_Interest`	The cumulative interest payment made up to and including this period.
`Balance`	The remaining mortgage balance at the end of the period.

Retirement Calculator

When planning for retirement, investors need to determine how much is needed and how to bridge any gaps in their financial plan. After collecting information about the user's financial situation and desired retirement lifestyle, this tool solves for three different parameters: deposit amount, retirement lifestyle, and retirement expenses.

Deposit Amount

A calculator to help plan for the deposit amount needed to retire comfortably.

HTTP REQUEST

```
POST /retirement_calculator/deposit_amount
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
```

NAV

```
        "current_age": 70,
        "death_age": 78,
        "retirement_expenses": 40000,
        "portfolio_return": 0.06,
        "retirement_age":75,
        "percent_of_expenses_covered": 0.25,
        "retirement_savings": 5000,
        "retirement_income": 5000,
        "deposit_schedule": {
           "deposit_frequency_interval": "year",
           "adjust_deposit_for_inflation": true
        },
        "inflation_rate": 0.02,
        "retirement_tax": 0.25
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/retirement_calculator/de
```

ARGUMENTS

Parameter	Description
`current_age` *integer, required*	The user's current age. This will be used when determining the amount of years to accumulate funds.
`death_age` *integer, required*	The user's death age. This age will be used when determining the amount of years that `retirement_expenses` will be needed.
`retirement_expenses` *float, required*	The desired annual living expenses or lifestyle in retirement.
`portfolio_return` *float, required*	The portfolio's annualized return.
`retirement_age` *integer*	The age the user will retire. This is the age that the user will cease adding to their retirement assets and begin drawing down funds to meet `retirement_expenses`. If excluded, defaults to `65`.
`percent_of_expenses_covered` *float*	The percentage of current expenses needed in retirement. If excluded, defaults to `0.70`.
`retirement_savings` *float*	The current amount of retirement savings. If excluded, defaults to `0`.

NAV

Parameter	Description
`retirement_income` *float*	The amount of an income stream that will be available in retirement, expressed in today's dollars. If excluded, defaults to `0`.
`deposit_schedule` *map*	The deposit schedule for the retirement goal.
`deposit_frequency_interval` *string*	The frequency at which to make additions to `retirement_savings` in the periods before retirement. The value may be one of the following: `year`, `quarter`, `month`, or `week`. If excluded, defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, the `deposit_amount` will be increased by the `inflation_rate`. If excluded, defaults to `true`.
`inflation_rate` *float*	The annual inflation rate of `retirement_expenses`. If excluded, defaults to `0`.
`retirement_tax` *float*	The tax on assets in retirement. This tax rate will be applied to the assets that are needed for `retirement_expenses`. If excluded, defaults to `0`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `retirement_savings`. If `retirement_savings` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `retirement_savings`. If `retirement_savings` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

Example Response

```
{
    "deposit_amount": 2351.62,
    "deposit_frequency_interval": "year",
    "projected_savings_at_retirement": 20456.61,
    "total_earnings": 5738.8,
```

NAV


```
        "total_contributions": 12237.92,
        "total_withdrawals": 17232.54,
        "total_taxes": 5744.18,
        "return_details": {
            "0": {
                "period_earnings": 0,
                "period_contribution": 0,
                "period_withdrawal": 0,
                "period_taxes": 0,
                "cumulative_earnings": 0,
                "cumulative_contributions": 0,
                "cumulative_withdrawals": 0,
                "cumulative_taxes": 0,
                "ending_balance": 5000
            },
            "1": {
                "period_earnings": 300,
                "period_contribution": 2351.62,
                "period_withdrawal": 0,
                "period_taxes": 0,
                "cumulative_earnings": 300,
                "cumulative_contributions": 2351.62,
                "cumulative_withdrawals": 0,
                "cumulative_taxes": 0,
                "ending_balance": 7651.62
            },
            "2": {
                "period_earnings": 459.1,
                "period_contribution": 2398.65,
                "period_withdrawal": 0,
                "period_taxes": 0,
                "cumulative_earnings": 759.1,
                "cumulative_contributions": 4750.27,
                "cumulative_withdrawals": 0,
                "cumulative_taxes": 0,
                "ending_balance": 10509.37
            },
            "3": {
                "period_earnings": 630.56,
                "period_contribution": 2446.62,
                "period_withdrawal": 0,
                "period_taxes": 0,
                "cumulative_earnings": 1389.66,
                "cumulative_contributions": 7196.89,
                "cumulative_withdrawals": 0,
                "cumulative_taxes": 0,
```

NAV III


```
            "ending_balance": 13586.55
        },
        "4": {
            "period_earnings": 815.19,
            "period_contribution": 2495.56,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 2204.85,
            "cumulative_contributions": 9692.45,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 16897.3
        },
        "5": {
            "period_earnings": 1013.84,
            "period_contribution": 2545.47,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 3218.69,
            "cumulative_contributions": 12237.92,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 20456.61
        },
        "6": {
            "period_earnings": 1227.4,
            "period_contribution": 0,
            "period_withdrawal": -5630.81,
            "period_taxes": -1876.94,
            "cumulative_earnings": 4446.09,
            "cumulative_contributions": 12237.92,
            "cumulative_withdrawals": -5630.81,
            "cumulative_taxes": -1876.94,
            "ending_balance": 14176.26
        },
        "7": {
            "period_earnings": 850.58,
            "period_contribution": 0,
            "period_withdrawal": -5743.43,
            "period_taxes": -1914.48,
            "cumulative_earnings": 5296.66,
            "cumulative_contributions": 12237.92,
            "cumulative_withdrawals": -11374.24,
            "cumulative_taxes": -3791.41,
            "ending_balance": 7368.93
        },
```

NAV


```
            "8": {
                "period_earnings": 442.14,
                "period_contribution": 0,
                "period_withdrawal": -5858.3,
                "period_taxes": -1952.77,
                "cumulative_earnings": 5738.8,
                "cumulative_contributions": 12237.92,
                "cumulative_withdrawals": -17232.54,
                "cumulative_taxes": -5744.18,
                "ending_balance": 0
            }
        }
    }
```


RESPONSE

Field	Description
`deposit_amount`	The amount to deposit in order to meet the retirement goal.
`deposit_frequency_interval`	The frequency interval of the deposit.
`projected_savings_at_retirement`	The total amount of savings projected to be available at retirement, expressed in today's dollars.
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`total_withdrawals`	The total amount of withdrawals taken over `decumulation_horizon`.
`total_taxes`	The total taxes paid on withdrawals over `decumulation_horizon`.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.

NAV

Field	Description
`period_contribution`	The deposit made for this period.
`period_withdrawal`	The withdrawal made for this period.
`period_taxes`	The taxes paid during this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`cumulative_taxes`	The cumulative taxes paid up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

Percent of Costs Covered

A calculator to help determine the percentage of the retirement lifestyle that can be afforded.

HTTP REQUEST

```
POST /retirement_calculator/percent_covered
```

NAV

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "current_age": 50,
        "death_age": 57,
        "retirement_expenses": 40000,
        "portfolio_return": 0.06,
        "retirement_age": 55,
        "retirement_savings": 5000,
        "retirement_income": 5000,
        "deposit_schedule": {
           "deposit_amount": 10000,
           "deposit_frequency_interval": "year",
           "adjust_deposit_for_inflation": true
        },
        "inflation_rate": 0.02,
        "retirement_tax": 0.25
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/retirement_calculator/pe
```

ARGUMENTS

Parameter	Description
`current_age` *integer, required*	The user's current age. This will be used when determining the amount of years to accumulate funds.
`death_age` *integer, required*	The user's death age. This age will be used when determining the amount of years that `retirement_expenses` will be needed.
`retirement_expenses` *float, required*	The desired annual living expenses or lifestyle in retirement.
`portfolio_return` *float, required*	The portfolio's annualized return.
`retirement_age` *integer*	The age the user will retire. This is the age that the user will cease adding to their retirement assets and begin drawing down

NAV

Parameter	Description
	funds to meet `retirement_expenses`. If excluded, defaults to `65`.
`retirement_savings` *float*	The current amount of retirement savings. If excluded, defaults to `0`.
`retirement_income` *float*	The amount of an income stream that will be available in retirement, expressed in today's dollars. If excluded, defaults to `0`.
`deposit_schedule` *map*	The deposit schedule for the retirement goal.
`deposit_amount` *float*	The periodic additions to retirement savings in the period before retirement. If excluded, defaults to `0`.
`deposit_frequency_interval` *string*	The frequency at which to make additions to retirement savings in the periods before retirement. The value may be one of the following: `year`, `quarter`, `month`, or `week`. If excluded, defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, the `deposit_amount` will be increased by the `inflation_rate`. If excluded, defaults to `true`.
`inflation_rate` *float*	The annual inflation rate of `retirement_expenses`. If excluded, defaults to `0`.
`retirement_tax` *float*	The tax on assets in retirement. This tax rate will be applied to the assets that are needed for `retirement_expenses`. If excluded, defaults to `0`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `retirement_savings`. If `retirement_savings` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `retirement_savings`. If `retirement_savings` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

NAV III

Example Response


```
{
    "achievable_expenses": 23465.95,
    "target_expenses": 35000,
    "percent_of_expenses_covered": 0.6705,
    "projected_savings_at_retirement": 65227.32,
    "total_earnings": 14134.9,
    "total_contributions": 52040.4,
    "total_withdrawals": 53381.48,
    "total_taxes": 17793.83,
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_contribution": 0,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 0,
            "cumulative_contributions": 0,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 5000
        },
        "1": {
            "period_earnings": 300,
            "period_contribution": 10000,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 300,
            "cumulative_contributions": 10000,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 15300
        },
        "2": {
            "period_earnings": 918,
            "period_contribution": 10200,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 1218,
            "cumulative_contributions": 20200,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 26418
```

NAV III


```
        },
        "3": {
            "period_earnings": 1585.08,
            "period_contribution": 10404,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 2803.08,
            "cumulative_contributions": 30604,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 38407.08
        },
        "4": {
            "period_earnings": 2304.42,
            "period_contribution": 10612.08,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 5107.5,
            "cumulative_contributions": 41216.08,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 51323.58
        },
        "5": {
            "period_earnings": 3079.42,
            "period_contribution": 10824.32,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 8186.92,
            "cumulative_contributions": 52040.4,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 65227.32
        },
        "6": {
            "period_earnings": 3913.64,
            "period_contribution": 0,
            "period_withdrawal": -26426.47,
            "period_taxes": -8808.82,
            "cumulative_earnings": 12100.56,
            "cumulative_contributions": 52040.4,
            "cumulative_withdrawals": -26426.47,
            "cumulative_taxes": -8808.82,
            "ending_balance": 33905.66
        },
        "7": {
```

NAV


```
                "period_earnings": 2034.34,
                "period_contribution": 0,
                "period_withdrawal": -26955,
                "period_taxes": -8985,
                "cumulative_earnings": 14134.9,
                "cumulative_contributions": 52040.4,
                "cumulative_withdrawals": -53381.48,
                "cumulative_taxes": -17793.83,
                "ending_balance": 0
            }
        }
    }
```


RESPONSE

Field	Description
`achievable_expenses`	The annual living expenses or lifestyle that can be covered in retirement, expressed in today's dollars.
`target_expenses`	The `retirement_expenses` input representing the target annual goal amount.
`percent_of_expenses_covered`	The percentage of `target_expenses` that can be covered.
`projected_savings_at_retirement`	The total amount of savings that are projected to be available at retirement, expressed in today's dollars.
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`total_withdrawals`	The total amount of withdrawals taken over `decumulation_horizon`.
`total_taxes`	The total taxes paid on withdrawals over `decumulation_horizon`.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or

NAV

Field	Description
	withdrawals are made.
`period_contribution`	The deposit made for this period.
`period_withdrawal`	The withdrawal made for this period.
`period_taxes`	The taxes paid during this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`cumulative_taxes`	The cumulative taxes paid up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

Retirement Expenses

A calculator to help plan for the amount of expenses that will be incurred in retirement.

HTTP REQUEST

```
POST /retirement_calculator/expenses
```

NAV

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "current_age": 50,
        "death_age": 57,
        "portfolio_return": 0.06,
        "retirement_age": 55,
        "percent_of_expenses_covered": 0.25,
        "retirement_savings": 5000,
        "retirement_income": 5000,
        "deposit_schedule": {
           "deposit_amount": 10000,
           "deposit_frequency_interval": "year",
           "adjust_deposit_for_inflation": true
        },
        "inflation_rate": 0.02,
        "retirement_tax": 0.25
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/retirement_calculator/ex
```

ARGUMENTS

Parameter	Description
`current_age` *integer, required*	The user's current age. This will be used when determining the amount of years to accumulate funds.
`death_age` *integer, required*	The user's death age. This age will be used when determining the amount of years that `retirement_expenses` will be needed.
`portfolio_return` *float, required*	The portfolio's annualized return.
`retirement_age` *integer*	The age the user will retire. This is the age that the user will cease adding to their retirement assets and begin drawing down funds to meet `retirement_expenses`. If excluded, defaults to `65`.

NAV

Parameter	Description
`percent_of_expenses_covered` *float*	The percentage of current expenses needed in retirement. If excluded, defaults to `0.70`.
`retirement_savings` *float*	The current amount of retirement savings. If excluded, defaults to `0`.
`retirement_income` *float*	The amount of an income stream that will be available in retirement, expressed in today's dollars. If excluded, defaults to `0`.
`deposit_schedule` *map*	The deposit schedule for the retirement goal.
`deposit_amount` *float*	The periodic additions to retirement savings in the period before retirement. If excluded, defaults to `0`.
`deposit_frequency_interval` *string*	The frequency at which to make additions to retirement savings in the periods before retirement. The value may be one of the following: `year`, `quarter`, `month`, or `week`. If excluded, defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, the `deposit_amount` will be increased by the `inflation_rate`. If excluded, defaults to `true`.
`inflation_rate` *float*	The annual inflation rate of `retirement_expenses`. If excluded, defaults to `0`.
`retirement_tax` *float*	The tax on assets in retirement. This tax rate will be applied to the assets that are needed for `retirement_expenses`. If excluded, defaults to `0`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `retirement_savings`. If `retirement_savings` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `retirement_savings`. If `retirement_savings` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

NAV III

Example Response


```
{
    "projected_retirement_expenses": 111336.69,
    "projected_retirement_expenses_adjusted": 98863.8,
    "projected_savings_at_retirement": 65227.32,
    "total_earnings": 14134.9,
    "total_contributions": 52040.4,
    "total_withdrawals": 53381.48,
    "total_taxes": 17793.83,
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_contribution": 0,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 0,
            "cumulative_contributions": 0,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 5000
        },
        "1": {
            "period_earnings": 300,
            "period_contribution": 10000,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 300,
            "cumulative_contributions": 10000,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 15300
        },
        "2": {
            "period_earnings": 918,
            "period_contribution": 10200,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 1218,
            "cumulative_contributions": 20200,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 26418
        },
```

NAV III


```
        "3": {
            "period_earnings": 1585.08,
            "period_contribution": 10404,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 2803.08,
            "cumulative_contributions": 30604,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 38407.08
        },
        "4": {
            "period_earnings": 2304.42,
            "period_contribution": 10612.08,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 5107.5,
            "cumulative_contributions": 41216.08,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 51323.58
        },
        "5": {
            "period_earnings": 3079.42,
            "period_contribution": 10824.32,
            "period_withdrawal": 0,
            "period_taxes": 0,
            "cumulative_earnings": 8186.92,
            "cumulative_contributions": 52040.4,
            "cumulative_withdrawals": 0,
            "cumulative_taxes": 0,
            "ending_balance": 65227.32
        },
        "6": {
            "period_earnings": 3913.64,
            "period_contribution": 0,
            "period_withdrawal": -26426.47,
            "period_taxes": -8808.82,
            "cumulative_earnings": 12100.56,
            "cumulative_contributions": 52040.4,
            "cumulative_withdrawals": -26426.47,
            "cumulative_taxes": -8808.82,
            "ending_balance": 33905.66
        },
        "7": {
            "period_earnings": 2034.34,
```

NAV


```
                "period_contribution": 0,
                "period_withdrawal": -26955,
                "period_taxes": -8985,
                "cumulative_earnings": 14134.9,
                "cumulative_contributions": 52040.4,
                "cumulative_withdrawals": -53381.48,
                "cumulative_taxes": -17793.83,
                "ending_balance": 0
            }
        }
    }
```


RESPONSE

Field	Description
`projected_retirement_expenses`	The after-tax annual living expenses or lifestyle available in retirement.
`projected_retirement_expenses_adjusted`	The after-tax annual living expenses or lifestyle available in retirement, expressed in today's dollars.
`projected_savings_at_retirement`	The total amount of savings that are projected to be available at retirement, expressed in today's dollars.
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`total_withdrawals`	The total amount of withdrawals taken over `decumulation_horizon`.
`total_taxes`	The total taxes paid on withdrawals over `decumulation_horizon`.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.

NAV

Field	Description
`period_contribution`	The deposit made for this period.
`period_withdrawal`	The withdrawal made for this period.
`period_taxes`	The taxes paid during this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`cumulative_taxes`	The cumulative taxes paid up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

Savings Calculator

While more detail is usually welcomed by financial professionals, some firms opt to provide simple calculators for simple savings or investing products. This endpoint provides a calculator that can show a simple growth of investment over time. After providing return, deposit, and balance information, the calculator returns an ending

balance and simulation details. Earnings are calculated at the beginning of each period, before deposits for that period are considered. Tax rates, if provided, are applied in the return calculation. Returns are calculated at the same frequency stipulated in `horizon_frequency_interval`.

HTTP REQUEST

`POST /savings_calculator`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "initial_balance": 10000,
        "horizon": 5,
        "return_schedule": [0.02, 0.025, 0.025, 0.025, 0.03],
        "horizon_frequency_interval": "year",
        "deposit_schedule":
        [
          {
            "deposit_amount": 100,
            "deposit_frequency_interval": "month",
            "deposit_duration": 120,
            "adjust_deposit_for_inflation": true
          },
          {
            "deposit_amount": 2000,
            "deposit_frequency_interval": "year",
            "deposit_duration": 10,
            "adjust_deposit_for_inflation": true
          }
        ],
        "tax_rate": 0.33,
        "inflation_rate": 0.02
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/savings_calculator"
```

ARGUMENTS

Parameter	Description
`horizon` *integer, required*	The number of time periods in the savings horizon, where each period represents `horizon_frequency_interval`.

NAV

Parameter	Description
`return_schedule` *array, required*	The rate of return per period. Providing an array of length 1, such as [0.04], will apply the value to each period in `horizon`.
`horizon_frequency_interval` *string*	The frequency interval for `horizon`. Must be one of `year`, `quarter`, `month`, `week`, or `day`. Defaults to `year`.
`initial_balance` *float*	The initial balance of the investment. Defaults to `0`.
`deposit_schedule` *array[map]*	Details on the deposit plan. If excluded, no deposits are included in the calculation.
`deposit_amount` *float*	The amount deposited in a given period and frequency. If excluded, defaults to `0`.
`deposit_frequency_interval` *string*	The period interval to be used in relation to `deposit_duration`. Must be one of `year`, `quarter`, `month`, `week`, or `day`. Defaults to `year`.
`deposit_duration` *string*	The amount of intervals for the time period. If excluded, defaults to the number of intervals given in `horizon`.
`adjust_deposit_for_inflation` *boolean*	If `true`, adjusts the deposit using the `inflation_rate`. If excluded, defaults to `true`.
`tax_rate` *float*	The tax rate to be applied to investment earnings. If excluded, defaults to `0`.
`inflation_rate` *float*	The inflation rate to be applied to deposits. If excluded, defaults to `0`.
`account_ids` *array[UUID]*	The ID(s) of the Nucleus account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.
`aggregation_account_ids` *array[UUID]*	The ID(s) of the Nucleus aggregation account(s) used to derive `initial_balance`. If `initial_balance` is not provided, we will try to use `account_ids` and/or `aggregation_account_ids` to derive this value instead.

NAV

Parameter	Description
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

Example Response


```
{
    "ending_balance": 28465.18,
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_contribution": 0,
            "period_withdrawal": 0,
            "cumulative_earnings": 0,
            "cumulative_contributions": 0,
            "cumulative_withdrawals": 0,
            "ending_balance": 10000
        },
        "1": {
            "period_earnings": 134,
            "period_contribution": 3264,
            "period_withdrawal": 0,
            "cumulative_earnings": 134,
            "cumulative_contributions": 3264,
            "cumulative_withdrawals": 0,
            "ending_balance": 13398
        },
        "2": {
            "period_earnings": 224.42,
            "period_contribution": 3329.28,
            "period_withdrawal": 0,
            "cumulative_earnings": 358.42,
            "cumulative_contributions": 6593.28,
            "cumulative_withdrawals": 0,
            "ending_balance": 16951.7
        },
        "3": {
            "period_earnings": 283.94,
```

NAV III


```
                "period_contribution": 3395.87,
                "period_withdrawal": 0,
                "cumulative_earnings": 642.36,
                "cumulative_contributions": 9989.15,
                "cumulative_withdrawals": 0,
                "ending_balance": 20631.5
            },
            "4": {
                "period_earnings": 345.58,
                "period_contribution": 3463.78,
                "period_withdrawal": 0,
                "cumulative_earnings": 987.94,
                "cumulative_contributions": 13452.93,
                "cumulative_withdrawals": 0,
                "ending_balance": 24440.86
            },
            "5": {
                "period_earnings": 491.26,
                "period_contribution": 3533.06,
                "period_withdrawal": 0,
                "cumulative_earnings": 1479.2,
                "cumulative_contributions": 16985.99,
                "cumulative_withdrawals": 0,
                "ending_balance": 28465.18
            }
        }
    }
```


RESPONSE

Field	Description
`ending_balance`	The ending balance of the investment, represented in today's dollars.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`period_contribution`	The deposit made for this period.

NAV

Field	Description
`period_withdrawal`	The withdrawal made for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Aggregation Account`
- `Aggregation Account Balance`

Variable Annuity

When planning for retirement or other financial outcomes, this service projects a variable annuity plan that is tied to a group of underlying investments. The Monte Carlo-based projection can help to evaluate potential annuity results and get a better understanding of how a variable annuity product may behave through both the accumulation phase and the payout phase.

HTTP REQUEST

`POST /variable_annuity`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
```

NAV

```
        "portfolio_tickers": [
            "HD",
            "CVX",
            "JNJ"
        ],
        "portfolio_weights": [
            0.3,
            0.3,
            0.4
        ],
        "accumulation_horizon": 4,
        "decumulation_horizon": 3,
        "initial_balance": 25000,
        "frequency_interval": "year",
        "deposit_schedule": {
            "deposit_amount": 1000    ,
            "deposit_frequency_interval": "year",
            "adjust_deposit_for_inflation": true
        },
        "inflation_rate": 0.01,
        "tax_rate": 0.2,
        "annuitization_rate": 0.05,
        "guaranteed_rate_benefit": [
            {
                "start": 1,
                "end": 7,
                "min_rate": 0.0,
                "max_rate": 0.15
            }
        ],
        "guaranteed_accumulation_benefit": 40000,
        "n": 1000,
        "result_type": "median",
        "p": 50,
        "remove_outliers": true,
        "start_date": "2016-01-01",
        "end_date": "2018-01-01",
        "trading_days_per_year": 252,
        "use_proxy_data": false
}' "https://[sandbox][api].hydrogenplatform.com/proton/v1/variable_annuity"
```

ARGUMENTS

NAV

Parameter	Description
`portfolio_tickers` *array, conditional requirement*	Portfolio tickers, referencing securities defined in the Nucleus API.
`portfolio_weights` *array, conditional requirement*	Portfolio weights, corresponding to `portfolio_tickers`. Must sum to `1.0`. Must be the same length as portfolio_tickers.
`accumulation_horizon` *integer, required*	The number of time periods until the payout phase begins.
`decumulation_horizon` *integer, required*	The number of time periods in the payout or decumulation phase.
`frequency_interval` *string*	Unit of time associated with `accumulation_horizon`, `decumulation_horizon`, `start`, `end`, and annuity payments. Value may be one of: year, quarter, month, week, day.
`initial_balance` *float*	The starting balance in the annuity plan, prior to any ongoing contributions. Defaults to `0`.
`deposit_schedule` *map*	The schedule of ongoing deposits to be made during the `accumulation_horizon`. Includes the fields shown below.
`deposit_amount` *float*	The periodic value of the deposit. Must be greater than or equal to 0. Defaults to `0`.
`deposit_frequency_interval` *string*	The period interval to be used in relation to deposit_amount. Must be one of `year`, `quarter`, `month`, `week`, or `day`. Defaults to `year`.
`adjust_deposit_for_inflation` *boolean*	If `true`, `deposit_amount` will be adjusted over time based on `inflation_rate`. Defaults to `true`.
`inflation_rate` *float*	The annualized rate of inflation. Defaults to `0`.
`tax_rate` *float*	The tax rate applied to annuity payouts. Defaults to `0`.
`annuitization_rate` *float*	The discount rate used to calculate annuity payout amounts during `decumulation_horizon`. Defaults to `0`.
`guaranteed_rate_benefit`	Boundaries enforced on the plan's rate of return. Each underlying map includes the fields below.

NAV

Parameter	Description
array	
`start` *integer*	The starting period for the guarantee, where `1` indicates the first period in `accumulation_horizon`. Must be greater than or equal to 1. Defaults to `1`.
`end` *integer*	The ending period for the guarantee. Must be less than or equal to the sum of `accumulation_horizon` and `decumulation_horizon`. Defaults to the sum of `accumulation_horizon` and `decumulation_horizon`.
`min_rate` *float*	The minimum allowable rate. Must be less than or equal to `max_rate`.
`max_rate` *float*	The maximum allowable rate. Must be greater than or equal to `min_rate`.
`guaranteed_accumulation_benefit` *float*	A guaranteed lower bound for the plan balance at the end of `accumulation_horizon`.
`n` *integer*	The number of Monte Carlo simulations to run. Defaults to `1000`.
`result_type` *string*	The type of Monte Carlo result to consider. Must be one of `mean`, `median`, or `custom`. Defaults to `median`.
`p` *float*	A Monte Carlo result percentile to consider, applicable when `result_type` is `custom`. Must be between 0 and 100 inclusive. Defaults to `50`.
`remove_outliers` *boolean*	If `true`, remove outlying results. If `true`, outlier analysis is performed on a median absolute deviation (MAD) basis, at the 2.5 threshold. Defaults to `false`.
`start_date` *date*	Start date used for ticker price history. Defaults to the earliest common date among `portfolio_tickers` prices.
`end_date` *date*	End date used for ticker price history. Defaults to the latest common date among `portfolio_tickers` prices.
`trading_days_per_year` *integer*	The number of days per year for which a portfolio is subject to market fluctuation. Defaults to `252`.

NAV

Parameter	Description
`use_proxy_data` *boolean*	If `true`, incorporate proxy price data as defined at the Security level in the Nucleus API. Proxy data is merged with base security data to form a continuous price history. Defaults to `false`.
`market_data_source` *string*	Source of security price data to be used in the analysis. Value may be `nucleus` or `integration`. When value is `nucleus`, security price data is sourced from Nucleus Security Price. When value is `integration`, security price data is sourced from a configured Integration Market Data vendor. Defaults to `nucleus`.
`account_id` *UUID, conditional requirement*	The ID of a Nucleus account used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`aggregation_account_id` *UUID, conditional requirement*	The ID of a Nucleus aggregation account used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`allocation_id` *UUID, conditional requirement*	The ID of a Nucleus allocation used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`model_id` *UUID, conditional requirement*	The ID of a Nucleus model used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`portfolio_id` *UUID, conditional requirement*	The ID of a Nucleus portfolio used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.

NAV

Parameter	Description
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

Example Response


```
{
    "accumulation_balance": 46331.26,
    "total_earnings": 29635.08,
    "total_contributions": 4101.01,
    "total_withdrawals": 46988.87,
    "total_taxes": 11747.22,
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_contribution": 0,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 0,
            "cumulative_contributions": 0,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 25000
        },
        "1": {
            "period_earnings": 3750,
            "period_contribution": 1010,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 3750,
            "cumulative_contributions": 1010,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 29760
        },
        "2": {
            "period_earnings": 4244.4,
            "period_contribution": 1020.1,
```

NAV III


```
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 7994.4,
            "cumulative_contributions": 2030.1,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 35024.5
        },
        "3": {
            "period_earnings": 4120.74,
            "period_contribution": 1030.3,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 12115.14,
            "cumulative_contributions": 3060.4,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 40175.54
        },
        "4": {
            "period_earnings": 5115.12,
            "period_contribution": 1040.6,
            "period_withdrawal": 0,
            "period_tax": 0,
            "cumulative_earnings": 17230.26,
            "cumulative_contributions": 4101.01,
            "cumulative_withdrawals": 0,
            "cumulative_tax": 0,
            "ending_balance": 46331.26
        },
        "5": {
            "period_earnings": 5696.16,
            "period_contribution": 0,
            "period_withdrawal": 14556.13,
            "period_tax": 3639.03,
            "cumulative_earnings": 22926.42,
            "cumulative_contributions": 4101.01,
            "cumulative_withdrawals": 14556.13,
            "cumulative_tax": 3639.03,
            "ending_balance": 33832.27
        },
        "6": {
            "period_earnings": 4395.9,
            "period_contribution": 0,
            "period_withdrawal": 15664.22,
            "period_tax": 3916.06,
```

NAV


```
            "cumulative_earnings": 27322.31,
            "cumulative_contributions": 4101.01,
            "cumulative_withdrawals": 30220.35,
            "cumulative_tax": 7555.09,
            "ending_balance": 18647.88
        },
        "7": {
            "period_earnings": 2312.76,
            "period_contribution": 0,
            "period_withdrawal": 16768.52,
            "period_tax": 4192.13,
            "cumulative_earnings": 29635.08,
            "cumulative_contributions": 4101.01,
            "cumulative_withdrawals": 46988.87,
            "cumulative_tax": 11747.22,
            "ending_balance": 0
        }
    }
}
```


RESPONSE

Field	Description
`accumulation_balance`	The balance at the end of `accumulation_horizon`, before annuity payments occur.
`total_earnings`	The total earnings generated over the horizon.
`total_contributions`	The total contributions added over the horizon.
`total_withdrawals`	The total amount of annuity payments over `decumulation_horizon`.
`total_taxes`	The total taxes paid on annuity payments over `decumulation_horizon`.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions are made.

NAV

Field	Description
`period_contribution`	The deposit made for this period.
`period_withdrawal`	The withdrawal made for this period.
`period_tax`	The tax amount for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_contributions`	The cumulative deposits made up to and including this period.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`cumulative_tax`	The cumulative taxes up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Security`
- `Security Price`
- `Account`
- `Portfolio`
- `Portfolio Holding`
- `Model`
- `Model Holding`
- `Allocation`
- `Allocation Composition`
- `Aggregation Account`
- `Aggregation Account Holding`

NAV

Only one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, and `portfolio_id` should be provided. An error will be thrown if multiple Nucleus data entities are referenced in a single request.

Goals

Accumulation Goal Allocation

This service offers a framework to match an accumulation goal to an appropriate portfolio, with the ability to prioritize an investor's risk appetite or prioritize the goal achievement. Allocations are selected from a universe based upon best fit for the selected priority, drawing from pre-defined portfolios and dynamically generated portfolios. This framework also has built-in support for a goal recommendation engine, which recommends actions that may be taken to increase the expected goal achievement probability if a goal cannot be achieved in the initially presented allocation context.

HTTP REQUEST

Example Request

`POST /goal_accumulation/allocation`

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "allocation_method": "create",
        "allocation_priority": "goal",
        "allocations": [
            "406ec16e-ebae-4130-a6c0-1cacb72569a2",
            "740f6466-3300-4122-927d-4a691359fa32",
            "0a02058f-4046-48de-b3d2-aeefd6d4efac"
        ],
        "opt_config": {
            "tickers": ["KHC", "AGG", "AMZN", "CMCSA", "XOM", "GOOGL"],
            "min_assets": 6,
            "w_config": {
                "w_min_major": 0.05,
                "w_max_major": 1,
                "w_min_minor": 0.05,
```

NAV


```
                "w_max_minor": 0.1,
                "cash_amount": 0.0
            },
            "w_asset_config": {
                "US_Equities": 1.0,
                "Fixed_Income": 1.0,
                "Intl_Equities": 1.0,
                "EM_Equities": 1.0,
                "Commodities": 1.0
            },
            "sec_types": ["minor", "major", "minor", "minor", "minor", "minor"],
            "start_date": "2016-01-01",
            "end_date": "2017-01-01"
        },
        "risk_score": 77,
        "trading_days_per_year": 252,
        "horizon": 5,
        "horizon_frequency": "year",
        "deposit_config": [
            {
                "dep_start_reference": "a_start",
                "dep_start_period": 0,
                "dep_end_reference": "a_start",
                "dep_end_period": 3,
                "dep_amount": 2000,
                "dep_frequency": "year",
                "dep_inflation": 0.0
            }
        ],
        "goal_config": {
            "goal_amount": 25000,
            "goal_inflation": 0.0
        },
        "recommendation_config": {
            "recommend": true,
            "inv_min": 0,
            "inv_max": 1000,
            "dep_min": 0,
            "dep_max": 1000,
            "horizon_min": 1,
            "horizon_max": 10,
            "recommended_inflation": 0.0
        },
        "curr_inv": 10000,
        "recommend_type": "recurring",
        "conf_tgt": 0.9,
```

NAV

```
        "n": 1000,
        "remove_outliers": true,
        "thresh_type": "perc",
        "thresh": 0,
        "withdrawal_tax": 0.0,
        "adjust_for_compounding": false,
        "compounding_rate": 0.0,
        "use_proxy_data": false
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/goal_accumulation/alloca
```

ARGUMENTS

Parameter	Description
`allocation_method` *string, required*	The allocation universe source, may be either `create` or `select`. `create` constructs an allocation universe from a dynamically generated efficient frontier, based on `opt_config`. `select` derives an allocation universe from pre-defined allocations, as stipulated by `allocations`.
`allocation_priority` *string, required*	The priority to consider when deriving the appropriate allocation, may be `risk` or `goal`. `risk` finds a portfolio that corresponds to an investor's `risk_score` within the universe of potential portfolios. `goal` does not consider the investor's `risk_score`, but rather allocates to a portfolio that best achieves the goal.
`opt_config` *map, conditional requirement*	Portfolio optimization configuration, required if `allocation_method = create`. Information in `opt_config` refers to security data, which must be created in advance using the Nucleus endpoints `POST /security` and `POST /security_price`. `opt_config` includes the fields shown below.
`tickers` *array, required*	A list of tickers to consider during the optimization process, referencing securities defined in the Nucleus API. We recommend using no more than 25 tickers in a single optimization. Pre-screening a securities universe prior to calling the service will result in faster execution times as well as more meaningful results.
`min_assets` *integer, required*	The minimum number of portfolio assets, excluding the cash security (if applicable).
`w_config` *map, required*	Weight constraints for security types, including `w_min_major`, `w_max_major`, `w_min_minor`, `w_max_minor`, and `cash_amount`, which stipulate lower and upper bounds for `minor` and `major`

NAV

Parameter	Description
	securities, as well as a constant weight for `cash` securities. Minimums are joined with zero to form a binary constraint. This means that the minimum weight of securities of each type may either be 0 or the stipulated minimum. For example, a `w_min_major` constraint of 5% indicates that if a `major` security is chosen during the optimization, its weight will be either 0% or >= 5%.
`w_asset_config` *map*	Weight constraints for asset classes. Stipulates a maximum weight for the asset classes represented in `tickers`. If an asset class is represented in `tickers` but not found in `w_asset_config`, the weight for that asset class will not be constrained.
`sec_types` *array, required*	A type for each security in `tickers`. Values may be `major`, `minor`, and `cash`. `major` securities are intended to be securities with more lenient weight thresholds. `minor` securities are intended to be securities with tighter weight thresholds. `cash` securities are constrained to a constant weight. `major` and `minor` designations interact with `w_config` to set distinct weight constraints for different kinds of securities. A maximum of one security may be labelled with the `cash` type.
`start_date` *date*	Start date for historical prices in yyyy-mm-dd format.
`end_date` *date*	End date for historical prices in yyyy-mm-dd format.
`allocations` *array(UUID)*	List of `allocation_id`s in the Nucleus API to select from. As a pre-requisite, must have values for the `performance` and `volatility` arguments within the chosen `allocation_id`. Defaults to include all available allocations.
`curr_inv` *float, conditional requirement*	The current amount invested toward the goal.
`horizon` *integer, conditional requirement*	The goal horizon, with a frequency defined by `horizon_frequency`. For example, for a client that is looking to make a major purchase in 30 years, use `horizon` = 30.
`horizon_frequency` *string, conditional requirement*	The frequency in relation to the number defined in `horizon`, and at which to run the simulation. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`.

NAV

Parameter	Description
`goal_config` *map, required*	Information to configure the accumulation goal amount.
`goal_amount` *string, conditional requirement*	The target goal amount, in today's dollars.
`goal_inflation` *integer*	The annualized inflation rate for `goal_amount`. Defaults to 0.
`deposit_config` *array(map)*	The deposits to occur over the course of the goal horizon. `deposit_config` is an array of maps, with each map containing the fields shown below.
`dep_start_reference` *string*	The reference for the starting point of the deposit. May be `a_start` or `a_end`, which refer to the start of the accumulation phase and the end of the accumulation phase, respectively. Defaults to `a_start`.
`dep_start_period` *integer*	The starting period for the deposit, relative to `dep_start_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `dep_frequency`. Defaults to 0.
`dep_end_reference` *string*	The reference for the ending point of the deposit. May be `a_start` or `a_end`, which refer to the start of the accumulation phase and the end of the accumulation phase, respectively. Defaults to `a_end`.
`dep_end_period` *integer*	The ending period for the deposit, relative to `dep_end_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `dep_frequency`. Defaults to 0.
`dep_amount` *float*	The deposit amount, in today's dollars. Defaults to 0.
`dep_frequency` *string*	The frequency of the deposit. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`. Defaults to `year`.
`dep_inflation` *float*	The annualized inflation rate for the deposit. Defaults to 0.

NAV

Parameter	Description
`recommendation_config` *map*	Information to configure the goal recommendation engine, including the fields shown below.
`recommend` *boolean*	If `true`, generate a recommended action to improve goal probability. Recommendations are only generated if a goal is off-track. Defaults to `true`.
`inv_min` *float*	The lower bound for `one-time` deposit recommendations. Defaults to 0.
`inv_max` *float*	The upper bound for `one-time` deposit recommendations. Defaults to `goal_amount`.
`dep_min` *float*	The lower bound for `recurring` deposit recommendations. Defaults to 0.
`dep_max` *float*	The upper bound for `recurring` deposit recommendations. Defaults to `goal_amount` divided by `horizon`.
`horizon_min` *integer*	The minimum number of periods (defined by `horizon_frequency`) that the accumulation horizon may be extended when `recommend_type = horizon`. Defaults to 1.
`horizon_max` *integer*	The maximum number of periods (defined by `horizon_frequency`) that the accumulation horizon may be extended when `recommend_type = horizon`. Defaults to 64.
`recommended_inflation` *float*	The annualized inflation rate attributed to a recommended recurring deposit.
`recommend_type` *string*	The type of recommended action. Value may be `recurring`, `one-time`, `combo`, or `horizon`. `recurring` returns a recurring periodic deposit. `one-time` returns a one-time deposit at the current time (occurring at time-zero, and reflected by an effective increase in the initial balance). `combo` returns a combination of recurring and one-time deposits. `horizon` returns a new accumulation horizon for the goal. Defaults to `horizon`.
`conf_tgt` *float*	The confidence target for goal achievement. `conf_tgt` indicates the level of statistical confidence that applies to the goal analysis. In some cases, the stipulated `conf_tgt` may be impossible to achieve within the

NAV

Parameter	Description
	given goal parameters; in these cases, the service will attempt to come as close as possible to satisfying the confidence target. Defaults to 0.90.
`n` *integer*	The number of simulations to run in the monte carlo simulation. Larger `n` values will make goal simulation results more consistent, but increase the execution time of the service. It has a maximum allowable value of 10000. Defaults to 2000.
`remove_outliers` *boolean*	If `true`, remove outlying results. If `true`, outlier analysis is performed on a median absolute deviation (MAD) basis, at the 2.5 MAD threshold. Defaults to true.
`thresh_type` *string*	The goal deviation threshold type. Value may be `perc` or `amnt`. `perc` indicates a percentage-based goal deviation threshold. `amnt` indicates a dollar-based goal deviation threshold. Defaults to `perc`.
`thresh` *float*	The goal deviation threshold value, corresponding to `thresh_type`. Defaults to 0.
`withdrawal_tax` *float*	The tax rate to apply to all withdrawals. Defaults to 0.
`trading_days_per_year` *integer*	The number of days per year for which a portfolio is subject to market fluctuation. `trading_days_per_year` impacts goal simulation results when `horizon_frequency = day`. Defaults to 252.
`risk_score` *integer*	The investor's risk score from 0 to 100, where 0 indicates minimum risk appetite and 100 indicates maximum risk appetite.
`adjust_for_compounding` *boolean*	If `true`, adjust periodic deposit amounts for compounding based on `compounding_rate`. This applies when a deposit's `dep_frequency` is shorter than `horizon_frequency`. Defaults to `false`.
`compounding_rate` *float*	The annualized rate to use when approximating a compounding effect on deposits. This value must be defined and `adjust_for_compounding` must be `true` in order to activate compounding adjustment. Defaults to 0.
`use_proxy_data` *boolean*	If `true`, incorporate proxy price data as defined at the Security level in the Nucleus API. Proxy data is merged with base security data to form a continuous price history. Defaults to `false`.

NAV

Parameter	Description
`market_data_source` *string*	Source of security price data to be used in the analysis. Value may be `nucleus` or `integration`. When value is `nucleus`, security price data is sourced from Nucleus Security Price. When value is `integration`, security price data is sourced from a configured Integration Market Data vendor. Defaults to `nucleus`.
`client_id` *UUID, conditional requirement*	The ID of a Nucleus client used to derive one or more of the following: `curr_inv`, `horizon`, `horizon_frequency`, and `goal_config.goal_amount`. If one or more of these required parameters are not provided, both `client_id` and `goal_id` are required.
`goal_id` *UUID, conditional requirement*	The ID of a Nucleus goal used to derive one or more of the following: `curr_inv`, `horizon`, `horizon_frequency`, and `goal_config.goal_amount`. If one or more of these required parameters are not provided, both `client_id` and `goal_id` are required.
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

Example Response


```
{
    "current_status": {
        "on_track": false,
        "progress": 0.4,
        "goal_probability": 0.5937,
        "goal_achievement_score": 66,
        "return_details": {
            "0": {
                "period_earnings": 0,
                "period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 0,
                "cumulative_deposits": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 10000
```

NAV III


```
    },
    "1": {
        "period_earnings": 353.71,
        "period_deposit": 2000,
        "period_withdrawal": 0,
        "cumulative_earnings": 353.71,
        "cumulative_deposits": 2000,
        "cumulative_withdrawals": 0,
        "ending_balance": 12353.71
    },
    "2": {
        "period_earnings": 853.48,
        "period_deposit": 2000,
        "period_withdrawal": 0,
        "cumulative_earnings": 1207.19,
        "cumulative_deposits": 4000,
        "cumulative_withdrawals": 0,
        "ending_balance": 15207.19
    },
    "3": {
        "period_earnings": 1417.59,
        "period_deposit": 2000,
        "period_withdrawal": 0,
        "cumulative_earnings": 2624.78,
        "cumulative_deposits": 6000,
        "cumulative_withdrawals": 0,
        "ending_balance": 18624.78
    },
    "4": {
        "period_earnings": 1569.72,
        "period_deposit": 0,
        "period_withdrawal": 0,
        "cumulative_earnings": 4194.5,
        "cumulative_deposits": 6000,
        "cumulative_withdrawals": 0,
        "ending_balance": 20194.5
    },
    "5": {
        "period_earnings": 1914.63,
        "period_deposit": 0,
        "period_withdrawal": 0,
        "cumulative_earnings": 6109.14,
        "cumulative_deposits": 6000,
        "cumulative_withdrawals": 0,
        "ending_balance": 22109.14
    }
```

NAV


```
            },
            "adjusted_goal_amount": 25000,
            "final_balance": 22109.14
        },
        "recommended_status": {
            "on_track": true,
            "progress": 0.4,
            "goal_probability": 0.9019,
            "goal_achievement_score": 100,
            "action": [
                {
                    "value": 500,
                    "freq_unit": "year",
                    "value_type": "recurring_deposit"
                }
            ],
            "return_details": {
                "0": {
                    "period_earnings": 0,
                    "period_deposit": 0,
                    "period_withdrawal": 0,
                    "cumulative_earnings": 0,
                    "cumulative_deposits": 0,
                    "cumulative_withdrawals": 0,
                    "ending_balance": 10000
                },
                "1": {
                    "period_earnings": 382.47,
                    "period_deposit": 2500,
                    "period_withdrawal": 0,
                    "cumulative_earnings": 382.47,
                    "cumulative_deposits": 2500,
                    "cumulative_withdrawals": 0,
                    "ending_balance": 12882.47
                },
                "2": {
                    "period_earnings": 957.58,
                    "period_deposit": 2500,
                    "period_withdrawal": 0,
                    "cumulative_earnings": 1340.05,
                    "cumulative_deposits": 5000,
                    "cumulative_withdrawals": 0,
                    "ending_balance": 16340.05
                },
                "3": {
                    "period_earnings": 1404.76,
```

NAV III


```
                "period_deposit": 2500,
                "period_withdrawal": 0,
                "cumulative_earnings": 2744.81,
                "cumulative_deposits": 7500,
                "cumulative_withdrawals": 0,
                "ending_balance": 20244.81
            },
            "4": {
                "period_earnings": 1802.08,
                "period_deposit": 500,
                "period_withdrawal": 0,
                "cumulative_earnings": 4546.89,
                "cumulative_deposits": 8000,
                "cumulative_withdrawals": 0,
                "ending_balance": 22546.89
            },
            "5": {
                "period_earnings": 1993.62,
                "period_deposit": 500,
                "period_withdrawal": 0,
                "cumulative_earnings": 6540.5,
                "cumulative_deposits": 8500,
                "cumulative_withdrawals": 0,
                "ending_balance": 25040.5
            }
        },
        "adjusted_goal_amount": 25000,
        "final_balance": 25040.5
    },
    "allocation": {
        "ret": 0.1177,
        "risk": 0.0665,
        "assets": [
            "AGG",
            "AMZN",
            "CMCSA",
            "GOOGL",
            "KHC",
            "XOM"
        ],
        "weights": [
            0.5,
            0.1,
            0.1,
            0.1,
            0.1,
```

NAV

```
                0.1
            ],
            "identifier": null
        }
    }
```

RESPONSE

Field	Description
`current_status`	The current status of the goal.
`on_track`	If `true`, the goal is on track.
`progress`	The goal progress percentage, defined as the current invested amount divided by the total target withdrawal amount over `d_horizon`.
`goal_probability`	The probability of achieving the goal with the given portfolio.
`goal_achievement_score`	A ratio of `goal_probability` to the `conf_tgt` on a scale from 0 to 100.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`period_deposit`	The deposit made for this period in the accumulation phase.
`period_withdrawal`	The withdrawal made for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_deposits`	The cumulative deposits made up to and including this period for the decumulation phase.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.

NAV

Field	Description
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.
`adjusted_goal_amount`	The effective goal target amount, adjusted for taxes, inflation, and goal deviation threshold.
`final_balance`	The projected final balance from `return_details`.
`recommended_status`	The goal status based on the provided recommendation.
`on_track`	If `true`, the goal is on track.
`progress`	The goal progress percentage, defined as the current invested amount divided by the total target withdrawal amount over `d_horizon`.
`goal_probability`	The probability of achieving the goal with the given portfolio.
`goal_achievement_score`	A ratio of `goal_probability` to the `conf_tgt` on a scale from 0 to 100.
`action`	The recommended action. Actions represent additions to the current configuration. For example, a recommended recurring deposit of 100 indicates 100 in addition to any existing deposits.
`value`	The value of the recommended action.
`freq_unit`	The frequency unit of `value`.
`value_type`	The type of recommendation being made.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`period_deposit`	The deposit made for this period in the accumulation phase.
`period_withdrawal`	The withdrawal made for this period.

NAV

Field	Description
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_deposits`	The cumulative deposits made up to and including this period for the decumulation phase.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.
`adjusted_goal_amount`	The effective goal target amount, adjusted for taxes, inflation, and goal deviation threshold.
`final_balance`	The projected final balance from `return_details`.
`allocation`	Information about the chosen portfolio, including risk and return.
`ret`	The portfolio annualized return.
`risk`	The portfolio annualized standard deviation.
`assets`	The securities in the created portfolio, returned if `allocation_method = create`.
`weights`	The weights for each of the securities in the created portfolio, returned if `allocation_method = create`.
`identifier`	The allocation's id, returned if `allocation_method = select`.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Security`
- `Security Price`
- `Client`
- `Goal`
- `Account`
- `Account Allocation`

NAV
- Allocation
- Allocation Composition
- Model
- Portfolio

Accumulation Goal Recommendation

This service is a recommendation engine that generates actions to increase the likelihood of achieving an accumulation goal, based on portfolio attributes and a target goal amount. Available actions include various types of cash inflows as well as goal horizon extension. The engine will attempt to find the minimal recommended action that satisfies a goal confidence target, subject to the given constraints.

HTTP REQUEST

```
POST /goal_accumulation/recommendation
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "p_ret": [0.09],
        "p_risk": [0.05],
        "trading_days_per_year": 252,
        "horizon": 5,
        "horizon_frequency": "year",
        "deposit_config": [
            {
                "dep_start_reference": "a_start",
                "dep_start_period": 0,
                "dep_end_reference": "a_start",
                "dep_end_period": 3,
                "dep_amount": 2000,
                "dep_frequency": "year",
                "dep_inflation": 0.0
            }
        ],
        "goal_config": {
            "goal_amount": 25000,
            "goal_inflation": 0.0
```

NAV

```
        },
        "recommendation_config": {
            "inv_min": 0,
            "inv_max": 1000,
            "dep_min": 0,
            "dep_max": 1000,
            "horizon_min": 1,
            "horizon_max": 10,
            "recommended_inflation": 0.0
        },
        "curr_inv": 10000,
        "recommend_type": "recurring",
        "conf_tgt": 0.9,
        "n": 1000,
        "remove_outliers": true,
        "thresh_type": "perc",
        "thresh": 0,
        "withdrawal_tax": 0.0,
        "adjust_for_compounding": false,
        "compounding_rate": 0.0
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/goal_accumulation/recomm
```

ARGUMENTS

Parameter	Description
`p_ret` *array, required*	The annual portfolio return per period. The length of the array must be less than or equal to `horizon`. If it is less, values will be conformed to the number of time periods by persisting the last value. For example, [0.04, 0.06] would result in an implied `p_ret` of [0.04, 0.06, 0.06, 0.06] for a simulation that spans four time periods.
`p_risk` *array, required*	The annual portfolio standard deviation per period. The length of the array must be less than or equal to `horizon`. If it is less, values will be conformed to the number of time periods by persisting the last value. For example, [0.04, 0.06] would result in an implied `p_risk` of [0.04, 0.06, 0.06, 0.06] for a simulation that spans four time periods.
`curr_inv` *float, conditional requirement*	The current amount invested toward the goal.
`horizon` *integer, conditional requirement*	The goal horizon, with a frequency defined by `horizon_frequency`. For example, for a client that is looking to make a major purchase in 30

NAV

Parameter	Description
	years, use `horizon` = 30.
`horizon_frequency` *string, conditional requirement*	The frequency in relation to the number defined in `horizon`, and at which to run the simulation. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`.
`goal_config` *map, required*	Information to configure the accumulation goal amount.
`goal_amount` *string, conditional requirement*	The target goal amount, in today's dollars.
`goal_inflation` *integer*	The annualized inflation rate for `goal_amount`. Defaults to 0.
`deposit_config` *array(map)*	The deposits to occur over the course of the goal horizon. `deposit_config` is an array of maps, with each map containing the fields shown below.
`dep_start_reference` *string*	The reference for the starting point of the deposit. May be `a_start` or `a_end`, which refer to the start of the accumulation phase and the end of the accumulation phase, respectively. Defaults to `a_start`.
`dep_start_period` *integer*	The starting period for the deposit, relative to `dep_start_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `dep_frequency`. Defaults to 0.
`dep_end_reference` *string*	The reference for the ending point of the deposit. May be `a_start` or `a_end`, which refer to the start of the accumulation phase and the end of the accumulation phase, respectively. Defaults to `a_end`.
`dep_end_period` *integer*	The ending period for the deposit, relative to `dep_end_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `dep_frequency`. Defaults to 0.
`dep_amount` *float*	The deposit amount, in today's dollars. Defaults to 0.

NAV

Parameter	Description
`dep_frequency` *string*	The frequency of the deposit. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`. Defaults to `year`.
`dep_inflation` *float*	The annualized inflation rate for the deposit. Defaults to 0.
`recommendation_config` *map*	Information to configure the goal recommendation engine, including the fields shown below.
`inv_min` *float*	The lower bound for `one-time` deposit recommendations. Defaults to 0.
`inv_max` *float*	The upper bound for `one-time` deposit recommendations. Defaults to `goal_amount`.
`dep_min` *float*	The lower bound for `recurring` deposit recommendations. Defaults to 0.
`dep_max` *float*	The upper bound for `recurring` deposit recommendations. Defaults to `goal_amount` divided by `horizon`.
`horizon_min` *integer*	The minimum number of periods (defined by `horizon_frequency`) that the accumulation horizon may be extended when `recommend_type = horizon`. Defaults to 1.
`horizon_max` *integer*	The maximum number of periods (defined by `horizon_frequency`) that the accumulation horizon may be extended when `recommend_type = horizon`. Defaults to 64.
`recommended_inflation` *float*	The annualized inflation rate attributed to a recommended recurring deposit.
`recommend_type` *string*	The type of recommended action. Value may be `recurring`, `one-time`, `combo`, or `horizon`. `recurring` returns a recurring periodic deposit. `one-time` returns a one-time deposit at the current time (occurring at time-zero, and reflected by an effective increase in the initial balance). `combo` returns a combination of recurring and one-time deposits. `horizon` returns a new accumulation horizon for the goal. Defaults to `horizon`.
`conf_tgt` *float*	The confidence target for goal achievement. `conf_tgt` indicates the level of statistical confidence that applies to the goal analysis. In some

NAV

Parameter	Description
	cases, the stipulated `conf_tgt` may be impossible to achieve within the given goal parameters; in these cases, the service will attempt to come as close as possible to satisfying the confidence target. Defaults to 0.90.
`n` *integer*	The number of simulations to run in the monte carlo simulation. Larger `n` values will make goal simulation results more consistent, but increase the execution time of the service. It has a maximum allowable value of 10000. Defaults to 2000.
`remove_outliers` *boolean*	If `true`, remove outlying results. If `true`, outlier analysis is performed on a median absolute deviation (MAD) basis, at the 2.5 MAD threshold. Defaults to true.
`thresh_type` *string*	The goal deviation threshold type. Value may be “perc” or “amnt”. `perc` indicates a percentage-based goal deviation threshold. `amnt` indicates a dollar-based goal deviation threshold. Defaults to `perc`.
`thresh` *float*	The goal deviation threshold value, corresponding to `thresh_type`. Defaults to 0.
`withdrawal_tax` *float*	The tax rate to apply to all withdrawals. Defaults to 0.
`trading_days_per_year` *integer*	The number of days per year for which a portfolio is subject to market fluctuation. `trading_days_per_year` impacts goal simulation results when `horizon_frequency = day`. Defaults to 252.
`adjust_for_compounding` *boolean*	If `true`, adjust periodic deposit amounts for compounding based on `compounding_rate`. This applies when a deposit’s `dep_frequency` is shorter than `horizon_frequency`. Defaults to `false`.
`compounding_rate` *float*	The annualized rate to use when approximating a compounding effect on deposits. This value must be defined and `adjust_for_compounding` must be `true` in order to activate compounding adjustment. Defaults to 0.
`client_id` *UUID, conditional requirement*	The ID of a Nucleus client used to derive one or more of the following: `curr_inv`, `horizon`, `horizon_frequency`, and `goal_config.goal_amount`. If one or more of these required parameters are not provided, both `client_id` and `goal_id` are required.

NAV

Parameter	Description
`goal_id` *UUID, conditional requirement*	The ID of a Nucleus goal used to derive one or more of the following: `curr_inv` , `horizon` , `horizon_frequency` , and `goal_config.goal_amount` . If one or more of these required parameters are not provided, both `client_id` and `goal_id` are required.
`create_log` *boolean*	If `true` , a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false` .

Example Response


```
{
    "on_track": true,
    "progress": 0.4,
    "goal_probability": 0.915,
    "goal_achievement_score": 100,
    "action": [
        {
            "value": 796.875,
            "freq_unit": "year",
            "value_type": "recurring_deposit"
        }
    ],
    "return_details": {
        "0": {
            "period_earnings": 0,
            "period_deposit": 0,
            "period_withdrawal": 0,
            "cumulative_earnings": 0,
            "cumulative_deposits": 0,
            "cumulative_withdrawals": 0,
            "ending_balance": 10000
        },
        "1": {
            "period_earnings": 271.88,
            "period_deposit": 2796.88,
            "period_withdrawal": 0,
            "cumulative_earnings": 271.88,
```

NAV III


```
                "cumulative_deposits": 2796.88,
                "cumulative_withdrawals": 0,
                "ending_balance": 13068.75
            },
            "2": {
                "period_earnings": 813.47,
                "period_deposit": 2796.88,
                "period_withdrawal": 0,
                "cumulative_earnings": 1085.35,
                "cumulative_deposits": 5593.75,
                "cumulative_withdrawals": 0,
                "ending_balance": 16679.1
            },
            "3": {
                "period_earnings": 1057.48,
                "period_deposit": 2796.88,
                "period_withdrawal": 0,
                "cumulative_earnings": 2142.83,
                "cumulative_deposits": 8390.62,
                "cumulative_withdrawals": 0,
                "ending_balance": 20533.46
            },
            "4": {
                "period_earnings": 1364.75,
                "period_deposit": 796.88,
                "period_withdrawal": 0,
                "cumulative_earnings": 3507.59,
                "cumulative_deposits": 9187.5,
                "cumulative_withdrawals": 0,
                "ending_balance": 22695.09
            },
            "5": {
                "period_earnings": 1735.93,
                "period_deposit": 796.88,
                "period_withdrawal": 0,
                "cumulative_earnings": 5243.52,
                "cumulative_deposits": 9984.38,
                "cumulative_withdrawals": 0,
                "ending_balance": 25227.89
            }
        },
        "adjusted_goal_amount": 25000,
        "final_balance": 25227.89
    }
```

NAV

RESPONSE

Field	Description
`on_track`	If `true`, the goal is on track.
`progress`	The goal progress percentage, defined as the current invested amount divided by the target `goal_amount`.
`goal_probability`	The probability of achieving the goal with the given portfolio.
`goal_achievement_score`	A ratio of `goal_probability` to the `conf_tgt` on a scale from 0 to 100.
`action`	The recommended action. Actions represent additions to the current configuration. For example, a recommended recurring deposit of 100 indicates 100 in addition to any existing deposits.
`value`	The value of the recommended action.
`freq_unit`	The frequency unit of `value`.
`value_type`	The type of recommendation being made.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`period_deposit`	The deposit made for this period in the accumulation phase.
`period_withdrawal`	The withdrawal made for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_deposit`	The cumulative deposits made up to and including this period for the accumulation phase.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NAV

Field	Description
`adjusted_goal_amount`	The effective goal target amount, adjusted for taxes, inflation, and goal deviation threshold.
`final_balance`	The projected final balance from `return_details`.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Goal`
- `Account`
- `Account Allocation`
- `Allocation`
- `Allocation Composition`
- `Model`
- `Portfolio`
- `Portfolio Asset Size`

Accumulation Goal Status

This service analyzes the state of expected goal achievement for an accumulation goal, based on the current goal configuration and context. In addition to a raw indication of whether a goal is on or off track, the service provides other metrics for advanced analysis. This framework also has built-in support for a goal recommendation engine, which recommends actions that may be taken to increase the expected goal achievement probability if a goal is off track.

HTTP REQUEST

```
POST /goal_accumulation/status
```

Example Request

NAV

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "p_ret": [0.09],
        "p_risk": [0.05],
        "trading_days_per_year": 252,
        "horizon": 5,
        "horizon_frequency": "year",
        "deposit_config": [
            {
                "dep_start_reference": "a_start",
                "dep_start_period": 0,
                "dep_end_reference": "a_start",
                "dep_end_period": 3,
                "dep_amount": 2000,
                "dep_frequency": "year",
                "dep_inflation": 0.0
            }
        ],
        "goal_config": {
            "goal_amount": 25000,
            "goal_inflation": 0.0
        },
        "recommendation_config": {
            "recommend": true,
            "inv_min": 0,
            "inv_max": 1000,
            "dep_min": 0,
            "dep_max": 1000,
            "horizon_min": 1,
            "horizon_max": 10,
            "recommended_inflation": 0.0
        },
        "curr_inv": 10000,
        "recommend_type": "recurring",
        "conf_tgt": 0.9,
        "n": 1000,
        "remove_outliers": true,
        "thresh_type": "perc",
        "thresh": 0,
        "withdrawal_tax": 0.0,
        "adjust_for_compounding": false,
```

NAV

```
        "compounding_rate": 0.0
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/goal_accumulation/status
```

ARGUMENTS

Parameter	Description
`p_ret` *array, required*	The annual portfolio return per period. The length of the array must be less than or equal to `horizon`. If it is less, values will be conformed to the number of time periods by persisting the last value. For example, [0.04, 0.06] would result in an implied `p_ret` of [0.04, 0.06, 0.06, 0.06] for a simulation that spans four time periods.
`p_risk` *array, required*	The annual portfolio standard deviation per period. The length of the array must be less than or equal to `horizon`. If it is less, values will be conformed to the number of time periods by persisting the last value. For example, [0.04, 0.06] would result in an implied `p_risk` of [0.04, 0.06, 0.06, 0.06] for a simulation that spans four time periods.
`curr_inv` *float, conditional requirement*	The current amount invested toward the goal.
`horizon` *integer, conditional requirement*	The goal horizon, with a frequency defined by `horizon_frequency`. For example, for a client that is looking to make a major purchase in 30 years, use `horizon` = 30.
`horizon_frequency` *string, conditional requirement*	The frequency in relation to the number defined in `horizon`, and at which to run the simulation. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`.
`goal_config` *map, required*	Information to configure the accumulation goal amount.
`goal_amount` *string, conditional requirement*	The target goal amount, in today's dollars.
`goal_inflation` *integer*	The annualized inflation rate for `goal_amount`. Defaults to 0.
`deposit_config` *array(map)*	The deposits to occur over the course of the goal horizon. `deposit_config` is an array of maps, with each map containing the

NAV

Parameter	Description
	fields shown below.
`dep_start_reference` *string*	The reference for the starting point of the deposit. May be `a_start` or `a_end`, which refer to the start of the accumulation phase and the end of the accumulation phase, respectively. Defaults to `a_start`.
`dep_start_period` *integer*	The starting period for the deposit, relative to `dep_start_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `dep_frequency`. Defaults to 0.
`dep_end_reference` *string*	The reference for the ending point of the deposit. May be `a_start` or `a_end`, which refer to the start of the accumulation phase and the end of the accumulation phase, respectively. Defaults to `a_end`.
`dep_end_period` *integer*	The ending period for the deposit, relative to `dep_end_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `dep_frequency`. Defaults to 0.
`dep_amount` *float*	The deposit amount, in today's dollars. Defaults to 0.
`dep_frequency` *string*	The frequency of the deposit. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`. Defaults to `year`.
`dep_inflation` *float*	The annualized inflation rate for the deposit. Defaults to 0.
`recommendation_config` *map*	Information to configure the goal recommendation engine, including the fields shown below.
`recommend` *boolean*	If `true`, generate a recommended action to improve goal probability. Recommendations are only generated if a goal is off-track. Defaults to `true`.
`inv_min` *float*	The lower bound for `one-time` deposit recommendations. Defaults to 0.
`inv_max`	The upper bound for `one-time` deposit recommendations. Defaults to `goal_amount`.

NAV

Parameter	Description
float	
`dep_min` *float*	The lower bound for `recurring` deposit recommendations. Defaults to 0.
`dep_max` *float*	The upper bound for `recurring` deposit recommendations. Defaults to `goal_amount` divided by `horizon`.
`horizon_min` *integer*	The minimum number of periods (defined by `horizon_frequency`) that the accumulation horizon may be extended when `recommend_type = horizon`. Defaults to 1.
`horizon_max` *integer*	The maximum number of periods (defined by `horizon_frequency`) that the accumulation horizon may be extended when `recommend_type = horizon`. Defaults to 64.
`recommended_inflation` *float*	The annualized inflation rate attributed to a recommended recurring deposit.
`recommend_type` *string*	The type of recommended action. Value may be `recurring`, `one-time`, `combo`, or `horizon`. `recurring` returns a recurring periodic deposit. `one-time` returns a one-time deposit at the current time (occurring at time-zero, and reflected by an effective increase in the initial balance). `combo` returns a combination of recurring and one-time deposits. `horizon` returns a new accumulation horizon for the goal. Defaults to `horizon`.
`conf_tgt` *float*	The confidence target for goal achievement. `conf_tgt` indicates the level of statistical confidence that applies to the goal analysis. In some cases, the stipulated `conf_tgt` may be impossible to achieve within the given goal parameters; in these cases, the service will attempt to come as close as possible to satisfying the confidence target. Defaults to 0.90.
`n` *integer*	The number of simulations to run in the monte carlo simulation. Larger `n` values will make goal simulation results more consistent, but increase the execution time of the service. It has a maximum allowable value of 10000. Defaults to 2000.
`remove_outliers` *boolean*	If `true`, remove outlying results. If `true`, outlier analysis is performed on a median absolute deviation (MAD) basis, at the 2.5 MAD threshold. Defaults to true.
`thresh_type` *string*	The goal deviation threshold type. Value may be “perc” or “amnt”. `perc` indicates a percentage-based goal deviation threshold. `amnt` indicates a

NAV

Parameter	Description
	dollar-based goal deviation threshold. Defaults to `perc`.
`thresh` *float*	The goal deviation threshold value, corresponding to `thresh_type`. Defaults to 0.
`withdrawal_tax` *float*	The tax rate to apply to all withdrawals. Defaults to 0.
`trading_days_per_year` *integer*	The number of days per year for which a portfolio is subject to market fluctuation. `trading_days_per_year` impacts goal simulation results when `horizon_frequency = day`. Defaults to 252.
`adjust_for_compounding` *boolean*	If `true`, adjust periodic deposit amounts for compounding based on `compounding_rate`. This applies when a deposit's `dep_frequency` is shorter than `horizon_frequency`. Defaults to `false`.
`compounding_rate` *float*	The annualized rate to use when approximating a compounding effect on deposits. This value must be defined and `adjust_for_compounding` must be `true` in order to activate compounding adjustment. Defaults to 0.
`client_id` *UUID, conditional requirement*	The ID of a Nucleus client used to derive one or more of the following: `curr_inv`, `horizon`, `horizon_frequency`, and `goal_config.goal_amount`. If one or more of these required parameters are not provided, both `client_id` and `goal_id` are required.
`goal_id` *UUID, conditional requirement*	The ID of a Nucleus goal used to derive one or more of the following: `curr_inv`, `horizon`, `horizon_frequency`, and `goal_config.goal_amount`. If one or more of these required parameters are not provided, both `client_id` and `goal_id` are required.
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

Example Response

NAV


```
{
    "current_status": {
        "on_track": false,
        "progress": 0.4,
        "goal_probability": 0.1638,
        "goal_achievement_score": 18,
        "return_details": {
            "0": {
                "period_earnings": 0,
                "period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 0,
                "cumulative_deposits": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 10000
            },
            "1": {
                "period_earnings": 274.26,
                "period_deposit": 2000,
                "period_withdrawal": 0,
                "cumulative_earnings": 274.26,
                "cumulative_deposits": 2000,
                "cumulative_withdrawals": 0,
                "ending_balance": 12274.26
            },
            "2": {
                "period_earnings": 665.94,
                "period_deposit": 2000,
                "period_withdrawal": 0,
                "cumulative_earnings": 940.19,
                "cumulative_deposits": 4000,
                "cumulative_withdrawals": 0,
                "ending_balance": 14940.19
            },
            "3": {
                "period_earnings": 954.05,
                "period_deposit": 2000,
                "period_withdrawal": 0,
                "cumulative_earnings": 1894.25,
                "cumulative_deposits": 6000,
                "cumulative_withdrawals": 0,
                "ending_balance": 17894.25
            },
            "4": {
                "period_earnings": 1068.64,
```

NAV ☰


```
                "period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 2962.89,
                "cumulative_deposits": 6000,
                "cumulative_withdrawals": 0,
                "ending_balance": 18962.89
            },
            "5": {
                "period_earnings": 1479.72,
                "period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 4442.61,
                "cumulative_deposits": 6000,
                "cumulative_withdrawals": 0,
                "ending_balance": 20442.61
            }
        },
        "adjusted_goal_amount": 25000,
        "final_balance": 20442.61
    },
    "recommended_status": {
        "on_track": true,
        "progress": 0.4,
        "goal_probability": 0.9025,
        "goal_achievement_score": 100,
        "action": [
            {
                "value": 781.25,
                "freq_unit": "year",
                "value_type": "recurring_deposit"
            }
        ],
        "return_details": {
            "0": {
                "period_earnings": 0,
                "period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 0,
                "cumulative_deposits": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 10000
            },
            "1": {
                "period_earnings": 302.11,
                "period_deposit": 2781.25,
                "period_withdrawal": 0,
```

NAV III


```
                "cumulative_earnings": 302.11,
                "cumulative_deposits": 2781.25,
                "cumulative_withdrawals": 0,
                "ending_balance": 13083.36
            },
            "2": {
                "period_earnings": 749.04,
                "period_deposit": 2781.25,
                "period_withdrawal": 0,
                "cumulative_earnings": 1051.14,
                "cumulative_deposits": 5562.5,
                "cumulative_withdrawals": 0,
                "ending_balance": 16613.64
            },
            "3": {
                "period_earnings": 1059.19,
                "period_deposit": 2781.25,
                "period_withdrawal": 0,
                "cumulative_earnings": 2110.34,
                "cumulative_deposits": 8343.75,
                "cumulative_withdrawals": 0,
                "ending_balance": 20454.09
            },
            "4": {
                "period_earnings": 1443.06,
                "period_deposit": 781.25,
                "period_withdrawal": 0,
                "cumulative_earnings": 3553.4,
                "cumulative_deposits": 9125,
                "cumulative_withdrawals": 0,
                "ending_balance": 22678.4
            },
            "5": {
                "period_earnings": 1579.8,
                "period_deposit": 781.25,
                "period_withdrawal": 0,
                "cumulative_earnings": 5133.2,
                "cumulative_deposits": 9906.25,
                "cumulative_withdrawals": 0,
                "ending_balance": 25039.45
            }
        },
        "adjusted_goal_amount": 25000,
        "final_balance": 25039.45
```

NAV

```
        }
    }
```

RESPONSE

Field	Description
`current_status`	The current status of the goal.
`on_track`	If `true`, the goal is on track.
`progress`	The goal progress percentage, defined as the current invested amount divided by the total target withdrawal amount over `d_horizon`.
`goal_probability`	The probability of achieving the goal with the given portfolio.
`goal_achievement_score`	A ratio of `goal_probability` to the `conf_tgt` on a scale from 0 to 100.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`period_deposit`	The deposit made for this period in the accumulation phase.
`period_withdrawal`	The withdrawal made for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`cumulative_deposit`	The cumulative deposits made up to and including this period for the decumulation phase.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NAV

Field	Description
`adjusted_goal_amount`	The effective goal target amount, adjusted for taxes, inflation, and goal deviation threshold.
`final_balance`	The projected final balance from `return_details`.
`recommended_status`	The goal status based on the provided recommendation.
`on_track`	If `true`, the goal is on track.
`progress`	The goal progress percentage, defined as the current invested amount divided by the total target withdrawal amount over `d_horizon`.
`goal_probability`	The probability of achieving the goal with the given portfolio.
`goal_achievement_score`	A ratio of `goal_probability` to the `conf_tgt` on a scale from 0 to 100.
`action`	The recommended action. Actions represent additions to the current configuration. For example, a recommended recurring deposit of 100 indicates 100 in addition to any existing deposits.
`value`	The value of the recommended action.
`freq_unit`	The frequency unit of `value`.
`value_type`	The type of recommendation being made.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`period_deposit`	The deposit made for this period in the accumulation phase.
`period_withdrawal`	The withdrawal made for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.

NAV

Field	Description
`cumulative_deposit`	The cumulative deposits made up to and including this period for the decumulation phase.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.
`adjusted_goal_amount`	The effective goal target amount, adjusted for taxes, inflation, and goal deviation threshold.
`final_balance`	The projected final balance from `return_details`.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Goal`
- `Account`
- `Account Allocation`
- `Allocation`
- `Allocation Composition`
- `Model`
- `Portfolio`
- `Portfolio Asset Size`

Decumulation Goal Allocation

This service offers a framework to match a decumulation goal to an appropriate portfolio, with the ability to prioritize an investor's risk appetite or prioritize the goal achievement. Allocations are selected from a universe based upon best fit for the selected priority, drawing from pre-defined portfolios and dynamically generated portfolios. This framework also has built-in support for a goal recommendation engine, which recommends actions that may be taken to increase the expected goal achievement probability if a goal cannot be achieved in the initially presented allocation context.

NAV

HTTP REQUEST

```
POST /goal_decumulation/allocation
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "allocation_method": "create",
        "allocation_priority": "goal",
        "allocations": [
            "406ec16e-ebae-4130-a6c0-1cacb72569a2",
            "740f6466-3300-4122-927d-4a691359fa32",
            "0a02058f-4046-48de-b3d2-aeefd6d4efac"
        ],
        "opt_config": {
            "tickers": ["KHC", "AGG", "AMZN", "CMCSA", "XOM", "GOOGL"],
            "min_assets": 6,
            "w_config": {
                "w_min_major": 0.05,
                "w_max_major": 1,
                "w_min_minor": 0.05,
                "w_max_minor": 0.1,
                "cash_amount": 0.0
            },
            "w_asset_config": {
                "US_Equities": 1.0,
                "Fixed_Income": 1.0,
                "Intl_Equities": 1.0,
                "EM_Equities": 1.0,
                "Commodities": 1.0
            },
            "sec_types": ["minor", "major","minor", "minor", "minor", "minor"],
            "start_date": "2016-01-01",
            "end_date": "2017-01-01"
        },
        "risk_score": 77,
        "trading_days_per_year": 252,
        "a_horizon": 3,
        "d_horizon": 2,
        "horizon_frequency": "year",
        "deposit_config": [
            {
                "dep_start_reference": "a_start",
```

NAV

```
                "dep_start_period": 0,
                "dep_end_reference": "a_start",
                "dep_end_period": 3,
                "dep_amount": 2000,
                "dep_frequency": "year",
                "dep_inflation": 0.0
            }
        ],
        "withdrawal_config": [
              {
                "with_amount": 11000,
                "with_start_reference": "a_end",
                "with_start_period": 0,
                "with_end_reference": "d_end",
                "with_end_period": 0,
                "with_frequency": "year",
                "with_inflation": 0.0
              }
        ],
        "recommendation_config": {
            "recommend": true,
            "inv_min": 0,
            "inv_max": 1000,
            "dep_min": 0,
            "dep_max": 1000,
            "horizon_min": 1,
            "horizon_max": 10,
            "recommended_inflation": 0.0
        },
        "curr_inv": 10000,
        "recommend_type": "recurring",
        "conf_tgt": 0.9,
        "n": 1000,
        "remove_outliers": true,
        "thresh_type": "perc",
        "thresh": 0,
        "withdrawal_tax": 0.0,
        "adjust_for_compounding": false,
        "compounding_rate": 0.0,
        "use_proxy_data": false
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/goal_decumulation/alloca
```

ARGUMENTS

NAV

Parameter	Description
`allocation_method` *string, required*	The allocation universe source, may be either `create` or `select`. `create` constructs an allocation universe from a dynamically generated efficient frontier, based on `opt_config`. `select` derives an allocation universe from pre-defined allocations, as stipulated by `allocations`.
`allocation_priority` *string, required*	The priority to consider when deriving the appropriate allocation, may be `risk` or `goal`. `risk` finds a portfolio that corresponds to an investor's `risk_score` within the universe of potential portfolios. `goal` does not consider the investor's `risk_score`, but rather allocates to a portfolio that best achieves the goal.
`opt_config` *map, conditional requirement*	Portfolio optimization configuration, required if `allocation_method = create`. Information in `opt_config` refers to security data, which must be created in advance using the Nucleus endpoints `POST /security` and `POST /security_price`. `opt_config` includes the fields shown below.
`tickers` *array, required*	A list of tickers to consider during the optimization process, referencing securities defined in the Nucleus API. We recommend using no more than 25 tickers in a single optimization. Pre-screening a securities universe prior to calling the service will result in faster execution times as well as more meaningful results.
`min_assets` *integer, required*	The minimum number of portfolio assets, excluding the cash security (if applicable).
`w_config` *map, required*	Weight constraints for security types, including `w_min_major`, `w_max_major`, `w_min_minor`, `w_max_minor`, and `cash_amount`, which stipulate lower and upper bounds for `minor` and `major` securities, as well as a constant weight for `cash` securities. Minimums are joined with zero to form a binary constraint. This means that the minimum weight of securities of each type may either be 0 or the stipulated minimum. For example, a `w_min_major` constraint of 5% indicates that if a `major` security is chosen during the optimization, its weight will be either 0% or >= 5%.
`w_asset_config` *map*	Weight constraints for asset classes. Stipulates a maximum weight for the asset classes represented in `tickers`. If an asset class is represented in `tickers` but not found in `w_asset_config`, the weight for that asset class will not be constrained.

NAV

Parameter	Description
`sec_types` *array, required*	A type for each security in `tickers`. Values may be `major`, `minor`, and `cash`. `major` securities are intended to be securities with more lenient weight thresholds. `minor` securities are intended to be securities with tighter weight thresholds. `cash` securities are constrained to a constant weight. `major` and `minor` designations interact with `w_config` to set distinct weight constraints for different kinds of securities. A maximum of one security may be labelled with the `cash` type.
`start_date` *date*	Start date for historical prices in yyyy-mm-dd format.
`end_date` *date*	End date for historical prices in yyyy-mm-dd format.
`allocations` *array(UUID)*	List of `allocation_id`s in the Nucleus API to select from. As a pre-requisite, must have values for the `performance` and `volatility` arguments within the chosen `allocation_id`. Defaults to include all available allocations.
`curr_inv` *float, conditional requirement*	The current amount invested toward the goal.
`a_horizon` *integer, conditional requirement*	The accumulation horizon, with a frequency defined by `horizon_frequency`. For example, for a client that is looking to retire in 30 years, use `a_horizon` = 30. Must be greater than 0 if `recommend_type` is `recurring` or `combo`.
`d_horizon` *integer, conditional requirement*	The decumulation horizon, with a frequency defined by `horizon_frequency`. For example, for a client that is looking to take withdrawals for 25 years during retirement, use `d_horizon` = 25. Must be greater than 0.
`horizon_frequency` *string, conditional requirement*	The frequency in relation to the numbers defined in `a_horizon` and `d_horizon`, and at which to run the simulation. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`.
`withdrawal_config` *array(map), conditional requirement*	The withdrawals to occur during the goal's decumulation phase. `withdrawal_config` is an array of maps, with each map containing the fields shown below.

NAV

Parameter	Description
`with_amount` *float, conditional requirement*	The withdrawal amount, in today's dollars.
`with_start_reference` *string*	The reference for the starting point of the withdrawal. May be `a_end` or `d_end`, which refer to the end of the accumulation phase and the end of the decumulation phase, respectively. Defaults to `a_end`.
`with_start_period` *integer*	The starting period for the withdrawal, relative to `with_start_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `with_frequency`. Defaults to 0.
`with_end_reference` *string*	The reference for the ending point of the withdrawal. May be `a_end` or `d_end`, which refer to the end of the accumulation phase and the end of the decumulation phase, respectively. Defaults to `d_end`.
`with_end_period` *integer*	The ending period for the withdrawal, relative to `with_end_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `with_frequency`. Defaults to 0.
`with_frequency` *string*	The frequency of the withdrawal. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`. Defaults to `year`.
`with_inflation` *float*	The annualized inflation rate for the withdrawal. Defaults to 0.
`deposit_config` *array(map)*	The deposits to occur over the course of the goal horizon. `deposit_config` is an array of maps, with each map containing the fields shown below.
`dep_start_reference` *string*	The reference for the starting point of the deposit. May be `a_start`, `a_end`, or `d_end`, which refer to the start of the accumulation phase, the end of the accumulation phase, and the end of the decumulation phase, respectively. Defaults to `a_start`.
`dep_start_period` *integer*	The starting period for the deposit, relative to `dep_start_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number

NAV

Parameter	Description
	indicates a period before the reference point. Periodicity depends upon `dep_frequency`. Defaults to 0.
`dep_end_reference` *string*	The reference for the ending point of the deposit. May be `a_start`, `a_end`, or `d_end`, which refer to the start of the accumulation phase, the end of the accumulation phase, and the end of the decumulation phase, respectively. Defaults to `a_end`.
`dep_end_period` *integer*	The ending period for the deposit, relative to `dep_end_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `dep_frequency`. Defaults to 0.
`dep_amount` *float*	The deposit amount, in today's dollars. Defaults to 0.
`dep_frequency` *string*	The frequency of the deposit. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`. Defaults to `year`.
`dep_inflation` *float*	The annualized inflation rate for the deposit. Defaults to 0.
`recommendation_config` *map*	Information to configure the goal recommendation engine, including the fields shown below.
`recommend` *boolean*	If `true`, generate a recommended action to improve goal probability. Recommendations are only generated if a goal is off-track. Defaults to `true`.
`inv_min` *float*	The lower bound for `one-time` deposit recommendations. Defaults to 0.
`inv_max` *float*	The upper bound for `one-time` deposit recommendations. Defaults to the first `with_amount` found in `withdrawal_config` multiplied by `d_horizon`.
`dep_min` *float*	The lower bound for `recurring` deposit recommendations. Defaults to 0.
`dep_max` *float*	The upper bound for `recurring` deposit recommendations. Defaults to the first `with_amount` found in `withdrawal_config` multiplied by

NAV

Parameter	Description
	`d_horizon` divided by `a_horizon`.
`horizon_min` *integer*	The minimum number of periods (defined by `horizon_frequency`) that the accumulation horizon may be extended when `recommend_type = horizon`. Defaults to 1.
`horizon_max` *integer*	The maximum number of periods (defined by `horizon_frequency`) that the accumulation horizon may be extended when `recommend_type = horizon`. Defaults to 64.
`recommended_inflation` *float*	The annualized inflation rate attributed to a recommended recurring deposit.
`recommend_type` *string*	The type of recommended action. Value may be `recurring`, `one-time`, `combo`, or `horizon`. `recurring` returns a recurring periodic deposit. `one-time` returns a one-time deposit at the current time (occurring at time-zero, and reflected by an effective increase in the initial balance). `combo` returns a combination of recurring and one-time deposits. `horizon` returns a new accumulation horizon for the goal. Defaults to `horizon`.
`conf_tgt` *float*	The confidence target for goal achievement. `conf_tgt` indicates the level of statistical confidence that applies to the goal analysis. In some cases, the stipulated `conf_tgt` may be impossible to achieve within the given goal parameters; in these cases, the service will attempt to come as close as possible to satisfying the confidence target. Defaults to 0.90.
`n` *integer*	The number of simulations to run in the monte carlo simulation. Larger `n` values will make goal simulation results more consistent, but increase the execution time of the service. It has a maximum allowable value of 10000. Defaults to 2000.
`remove_outliers` *boolean*	If `true`, remove outlying results. If `true`, outlier analysis is performed on a median absolute deviation (MAD) basis, at the 2.5 MAD threshold. Defaults to true.
`thresh_type` *string*	The goal deviation threshold type. Value may be “perc” or “amnt”. `perc` indicates a percentage-based goal deviation threshold. `amnt` indicates a dollar-based goal deviation threshold. Defaults to `perc`.
`thresh` *float*	The goal deviation threshold value, corresponding to `thresh_type`. Defaults to 0.

NAV

Parameter	Description
`withdrawal_tax` *float*	The tax rate to apply to all withdrawals. Defaults to 0.
`trading_days_per_year` *integer*	The number of days per year for which a portfolio is subject to market fluctuation. `trading_days_per_year` impacts goal simulation results when `horizon_frequency = day`. Defaults to 252.
`risk_score` *integer*	The investor's risk score from 0 to 100, where 0 indicates minimum risk appetite and 100 indicates maximum risk appetite.
`adjust_for_compounding` *boolean*	If `true`, adjust periodic deposit amounts for compounding based on `compounding_rate`. This applies when a deposit's `dep_frequency` is shorter than `horizon_frequency`. Defaults to `false`.
`compounding_rate` *float*	The annualized rate to use when approximating a compounding effect on deposits. This value must be defined and `adjust_for_compounding` must be `true` in order to activate compounding adjustment. Defaults to 0.
`use_proxy_data` *boolean*	If `true`, incorporate proxy price data as defined at the Security level in the Nucleus API. Proxy data is merged with base security data to form a continuous price history. Defaults to `false`.
`market_data_source` *string*	Source of security price data to be used in the analysis. Value may be `nucleus` or `integration`. When value is `nucleus`, security price data is sourced from Nucleus Security Price. When value is `integration`, security price data is sourced from a configured Integration Market Data vendor. Defaults to `nucleus`.
`client_id` *UUID, conditional requirement*	The ID of a Nucleus client used to derive one or more of the following: `curr_inv`, `a_horizon`, `d_horizon`, `horizon_frequency`, and `withdrawal_config.with_amount`. If one or more of these required parameters are not provided, both `client_id` and `goal_id` are required.
`goal_id` *UUID, conditional requirement*	The ID of a Nucleus goal used to derive one or more of the following: `curr_inv`, `a_horizon`, `d_horizon`, `horizon_frequency`, and `withdrawal_config.with_amount`. If one or more of these required parameters are not provided, both `client_id` and `goal_id` are required.

NAV

Parameter	Description
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

Example Response


```
{
    "current_status": {
        "on_track": false,
        "progress": 0.4545,
        "goal_probability": 0.8342,
        "goal_achievement_score": 93,
        "return_details": {
            "0": {
                "period_earnings": 0,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 0,
                "accumulation_cumulative_deposit": 0,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 10000
            },
            "1": {
                "period_earnings": 326.32,
                "accumulation_period_deposit": 2000,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 326.32,
                "accumulation_cumulative_deposit": 2000,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 12326.32
            },
            "2": {
                "period_earnings": 1033.7,
                "accumulation_period_deposit": 2000,
                "decumulation_period_deposit": 0,
```

NAV III


```
                "period_withdrawal": 0,
                "cumulative_earnings": 1360.02,
                "accumulation_cumulative_deposit": 4000,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 15360.02
            },
            "3": {
                "period_earnings": 1245.48,
                "accumulation_period_deposit": 2000,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 2605.5,
                "accumulation_cumulative_deposit": 6000,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 18605.5
            },
            "4": {
                "period_earnings": 1669.58,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 11000,
                "cumulative_earnings": 4275.09,
                "accumulation_cumulative_deposit": 6000,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 11000,
                "ending_balance": 9275.09
            },
            "5": {
                "period_earnings": 0,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 9275.09,
                "cumulative_earnings": 4275.09,
                "accumulation_cumulative_deposit": 6000,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 20275.09,
                "ending_balance": 0
            }
        },
        "accumulation_balance": 18605.5
    },
    "recommended_status": {
        "on_track": true,
        "progress": 0.4545,
```

NAV


```
        "goal_probability": 0.9065,
        "goal_achievement_score": 100,
        "action": [
            {
                "value": 218.75,
                "freq_unit": "year",
                "value_type": "recurring_deposit"
            }
        ],
        "return_details": {
            "0": {
                "period_earnings": 0,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 0,
                "accumulation_cumulative_deposit": 0,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 10000
            },
            "1": {
                "period_earnings": 355.11,
                "accumulation_period_deposit": 2218.75,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 355.11,
                "accumulation_cumulative_deposit": 2218.75,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 12573.86
            },
            "2": {
                "period_earnings": 1040.96,
                "accumulation_period_deposit": 2218.75,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 1396.07,
                "accumulation_cumulative_deposit": 4437.5,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 15833.57
            },
            "3": {
                "period_earnings": 1230.63,
                "accumulation_period_deposit": 2218.75,
```

NAV III


```
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 2626.7,
                "accumulation_cumulative_deposit": 6656.25,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 19282.95
            },
            "4": {
                "period_earnings": 1864.12,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 11000,
                "cumulative_earnings": 4490.82,
                "accumulation_cumulative_deposit": 6656.25,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 11000,
                "ending_balance": 10147.07
            },
            "5": {
                "period_earnings": 988.6,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 11000,
                "cumulative_earnings": 5479.43,
                "accumulation_cumulative_deposit": 6656.25,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 22000,
                "ending_balance": 135.68
            }
        },
        "accumulation_balance": 19282.95
    },
    "allocation": {
        "ret": 0.1177,
        "risk": 0.0665,
        "assets": [
            "AGG",
            "AMZN",
            "CMCSA",
            "GOOGL",
            "KHC",
            "XOM"
        ],
        "weights": [
            0.5,
```

NAV

```
                0.1,
                0.1,
                0.1,
                0.1,
                0.1
            ],
            "identifier": null
        }
    }
```

RESPONSE

Field	Description
`current_status`	The current status of the goal.
`on_track`	If `true` , the goal is on track.
`progress`	The goal progress percentage, defined as the current invested amount divided by the total target withdrawal amount over `d_horizon` .
`goal_probability`	The probability of achieving the goal with the given portfolio.
`goal_achievement_score`	A ratio of `goal_probability` to the `conf_tgt` on a scale from 0 to 100.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`accumulation_period_deposit`	The deposit made for this period in the accumulation phase.
`decumulation_period_deposit`	The deposit made for this period in the decumulation phase.
`period_withdrawal`	The withdrawal made for this period.

NAV

Field	Description
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`accumulation_cumulative_deposit`	The cumulative deposits made up to and including this period for the accumulation phase.
`decumulation_cumulative_deposit`	The cumulative deposits made up to and including this period for the decumulation phase.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.
`accumulation_balance`	The projected balance at the end of the accumulation horizon.
`recommended_status`	The goal status based on the provided recommendation.
`on_track`	If `true` , the goal is on track.
`progress`	The goal progress percentage, defined as the current invested amount divided by the total target withdrawal amount over `d_horizon` .
`goal_probability`	The probability of achieving the goal with the given portfolio.
`goal_achievement_score`	A ratio of `goal_probability` to the `conf_tgt` on a scale from 0 to 100.
`action`	The recommended action. Actions represent additions to the current configuration. For example, a recommended recurring deposit of 100 indicates 100 in addition to any existing deposits.
`value`	The value of the recommended action.
`freq_unit`	The frequency unit of `value` .
`value_type`	The type of recommendation being made.

NAV

Field	Description
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`accumulation_period_deposit`	The deposit made for this period in the accumulation phase.
`decumulation_period_deposit`	The deposit made for this period in the decumulation phase.
`period_withdrawal`	The withdrawal made for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`accumulation_cumulative_deposit`	The cumulative deposits made up to and including this period for the accumulation phase.
`decumulation_cumulative_deposit`	The cumulative deposits made up to and including this period for the decumulation phase.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.
`accumulation_balance`	The projected balance at the end of the accumulation horizon.
`allocation`	Information about the chosen portfolio, including risk and return.
`ret`	The portfolio annualized return.
`risk`	The portfolio annualized standard deviation.
`assets`	The securities in the created portfolio, returned if `allocation_method = create`.

NAV

Field	Description
`weights`	The weights for each of the securities in the created portfolio, returned if `allocation_method = create`.
`identifier`	The allocation's id, returned if `allocation_method = select`.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- Security
- `Security Price`
- `Client`
- `Goal`
- `Account`
- `Account Allocation`
- `Allocation`
- `Allocation Composition`
- `Model`
- `Portfolio`

Decumulation Goal Recommendation

This service is a recommendation engine that generates actions to increase the likelihood of achieving a decumulation goal, based on portfolio attributes and a target goal amount. Available actions include various types of cash inflows as well as goal horizon extension. The engine will attempt to find the minimal recommended action that satisfies a goal confidence target, subject to the given constraints.

HTTP REQUEST

```
POST /goal_decumulation/recommendation
```

Example Request

NAV

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "p_ret": [0.09],
        "p_risk": [0.05],
        "trading_days_per_year": 252,
        "a_horizon": 3,
        "d_horizon": 2,
        "horizon_frequency": "year",
        "deposit_config": [
            {
                "dep_start_reference": "a_start",
                "dep_start_period": 0,
                "dep_end_reference": "a_start",
                "dep_end_period": 3,
                "dep_amount": 2000,
                "dep_frequency": "year",
                "dep_inflation": 0.0
            }
        ],
        "withdrawal_config": [
            {
                "with_amount": 11000,
                "with_start_reference": "a_end",
                "with_start_period": 0,
                "with_end_reference": "d_end",
                "with_end_period": 0,
                "with_frequency": "year",
                "with_inflation": 0.0
            }
        ],
        "recommendation_config": {
            "inv_min": 0,
            "inv_max": 1000,
            "dep_min": 0,
            "dep_max": 1000,
            "horizon_min": 1,
            "horizon_max": 10,
            "recommended_inflation": 0.0
        },
        "curr_inv": 10000,
        "recommend_type": "recurring",
        "conf_tgt": 0.9,
        "n": 1000,
        "remove_outliers": true,
```

NAV

```
        "thresh_type": "perc",
        "thresh": 0,
        "withdrawal_tax": 0.0,
        "adjust_for_compounding": false,
        "compounding_rate": 0.0
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/goal_decumulation/recomm
```

ARGUMENTS

Parameter	Description
`p_ret` *array, required*	The annual portfolio return per period. The length of the array must be less than or equal to the sum of `a_horizon` and `d_horizon`. If it is less, values will be conformed to the number of time periods by persisting the last value. For example, [0.04, 0.06] would result in an implied `p_ret` of [0.04, 0.06, 0.06, 0.06] for a simulation that spans four time periods.
`p_risk` *array, required*	The annual portfolio standard deviation per period. The length of the array must be less than or equal to the sum of `a_horizon` and `d_horizon`. If it is less, values will be conformed to the number of time periods by persisting the last value. For example, [0.04, 0.06] would result in an implied `p_risk` of [0.04, 0.06, 0.06, 0.06] for a simulation that spans four time periods.
`curr_inv` *float, conditional requirement*	The current amount invested toward the goal.
`a_horizon` *integer, conditional requirement*	The accumulation horizon, with a frequency defined by `horizon_frequency`. For example, for a client that is looking to retire in 30 years, use `a_horizon` = 30. Must be greater than 0 if `recommend_type` is `recurring` or `combo`.
`d_horizon` *integer, conditional requirement*	The decumulation horizon, with a frequency defined by `horizon_frequency`. For example, for a client that is looking to take withdrawals for 25 years during retirement, use `d_horizon` = 25. Must be greater than 0.
`horizon_frequency` *string, conditional requirement*	The frequency in relation to the numbers defined in `a_horizon` and `d_horizon`, and at which to run the simulation. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`.

NAV

Parameter	Description
`withdrawal_config` *array(map), conditional requirement*	The withdrawals to occur during the goal's decumulation phase. `withdrawal_config` is an array of maps, with each map containing the fields shown below.
`with_amount` *float, conditional requirement*	The withdrawal amount, in today's dollars.
`with_start_reference` *string*	The reference for the starting point of the withdrawal. May be `a_end` or `d_end`, which refer to the end of the accumulation phase and the end of the decumulation phase, respectively. Defaults to `a_end`.
`with_start_period` *integer*	The starting period for the withdrawal, relative to `with_start_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `with_frequency`. Defaults to 0.
`with_end_reference` *string*	The reference for the ending point of the withdrawal. May be `a_end` or `d_end`, which refer to the end of the accumulation phase and the end of the decumulation phase, respectively. Defaults to `d_end`.
`with_end_period` *integer*	The ending period for the withdrawal, relative to `with_end_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `with_frequency`. Defaults to 0.
`with_frequency` *string*	The frequency of the withdrawal. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`. Defaults to `year`.
`with_inflation` *float*	The annualized inflation rate for the withdrawal. Defaults to 0.
`deposit_config` *array(map)*	The deposits to occur over the course of the goal horizon. `deposit_config` is an array of maps, with each map containing the fields shown below.
`dep_start_reference` *string*	The reference for the starting point of the deposit. May be `a_start`, `a_end`, or `d_end`, which refer to the start of the accumulation phase, the end of the accumulation phase, and the end of the decumulation phase, respectively. Defaults to `a_start`.

NAV

Parameter	Description
`dep_start_period` *integer*	The starting period for the deposit, relative to `dep_start_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `dep_frequency`. Defaults to 0.
`dep_end_reference` *string*	The reference for the ending point of the deposit. May be `a_start`, `a_end`, or `d_end`, which refer to the start of the accumulation phase, the end of the accumulation phase, and the end of the decumulation phase, respectively. Defaults to `a_end`.
`dep_end_period` *integer*	The ending period for the deposit, relative to `dep_end_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `dep_frequency`. Defaults to 0.
`dep_amount` *float*	The deposit amount, in today's dollars. Defaults to 0.
`dep_frequency` *string*	The frequency of the deposit. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`. Defaults to `year`.
`dep_inflation` *float*	The annualized inflation rate for the deposit. Defaults to 0.
`recommendation_config` *map*	Information to configure the goal recommendation engine, including the fields shown below.
`inv_min` *float*	The lower bound for `one-time` deposit recommendations. Defaults to 0.
`inv_max` *float*	The upper bound for `one-time` deposit recommendations. Defaults to the first `with_amount` found in `withdrawal_config` multiplied by `d_horizon`.
`dep_min` *float*	The lower bound for `recurring` deposit recommendations. Defaults to 0.
`dep_max` *float*	The upper bound for `recurring` deposit recommendations. Defaults to the first `with_amount` found in `withdrawal_config` multiplied by `d_horizon` divided by `a_horizon`.

NAV

Parameter	Description
`horizon_min` *integer*	The minimum number of periods (defined by `horizon_frequency`) that the accumulation horizon may be extended when `recommend_type = horizon`. Defaults to 1.
`horizon_max` *integer*	The maximum number of periods (defined by `horizon_frequency`) that the accumulation horizon may be extended when `recommend_type = horizon`. Defaults to 64.
`recommended_inflation` *float*	The annualized inflation rate attributed to a recommended recurring deposit.
`recommend_type` *string*	The type of recommended action. Value may be `recurring`, `one-time`, `combo`, or `horizon`. `recurring` returns a recurring periodic deposit. `one-time` returns a one-time deposit at the current time (occurring at time-zero, and reflected by an effective increase in the initial balance). `combo` returns a combination of recurring and one-time deposits. `horizon` returns a new accumulation horizon for the goal. Defaults to `horizon`.
`conf_tgt` *float*	The confidence target for goal achievement. `conf_tgt` indicates the level of statistical confidence that applies to the goal analysis. In some cases, the stipulated `conf_tgt` may be impossible to achieve within the given goal parameters; in these cases, the service will attempt to come as close as possible to satisfying the confidence target. Defaults to 0.90.
`n` *integer*	The number of simulations to run in the monte carlo simulation. Larger `n` values will make goal simulation results more consistent, but increase the execution time of the service. It has a maximum allowable value of 10000. Defaults to 2000.
`remove_outliers` *boolean*	If `true`, remove outlying results. If `true`, outlier analysis is performed on a median absolute deviation (MAD) basis, at the 2.5 MAD threshold. Defaults to true.
`thresh_type` *string*	The goal deviation threshold type. Value may be “perc” or “amnt”. `perc` indicates a percentage-based goal deviation threshold. `amnt` indicates a dollar-based goal deviation threshold. Defaults to `perc`.
`thresh` *float*	The goal deviation threshold value, corresponding to `thresh_type`. Defaults to 0.

NAV

Parameter	Description
`withdrawal_tax` *float*	The tax rate to apply to all withdrawals. Defaults to 0.
`trading_days_per_year` *integer*	The number of days per year for which a portfolio is subject to market fluctuation. `trading_days_per_year` impacts goal simulation results when `horizon_frequency = day`. Defaults to 252.
`adjust_for_compounding` *boolean*	If `true`, adjust periodic deposit amounts for compounding based on `compounding_rate`. This applies when a deposit's `dep_frequency` is shorter than `horizon_frequency`. Defaults to `false`.
`compounding_rate` *float*	The annualized rate to use when approximating a compounding effect on deposits. This value must be defined and `adjust_for_compounding` must be `true` in order to activate compounding adjustment. Defaults to 0.
`client_id` *UUID, conditional requirement*	The ID of a Nucleus client used to derive one or more of the following: `curr_inv`, `a_horizon`, `d_horizon`, `horizon_frequency`, and `withdrawal_config.with_amount`. If one or more of these required parameters are not provided, both `client_id` and `goal_id` are required.
`goal_id` *UUID, conditional requirement*	The ID of a Nucleus goal used to derive one or more of the following: `curr_inv`, `a_horizon`, `d_horizon`, `horizon_frequency`, and `withdrawal_config.with_amount`. If one or more of these required parameters are not provided, both `client_id` and `goal_id` are required.
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

Example Response

```
{
    "on_track": true,
    "progress": 0.4545,
    "goal_probability": 0.9127,
```

NAV


```
        "goal_achievement_score": 100,
        "action": [
            {
                "value": 625,
                "freq_unit": "year",
                "value_type": "recurring_deposit"
            }
        ],
        "return_details": {
            "0": {
                "period_earnings": 0,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 0,
                "accumulation_cumulative_deposit": 0,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 10000
            },
            "1": {
                "period_earnings": 261.03,
                "accumulation_period_deposit": 2625,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 261.03,
                "accumulation_cumulative_deposit": 2625,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 12886.03
            },
            "2": {
                "period_earnings": 819.01,
                "accumulation_period_deposit": 2625,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 1080.04,
                "accumulation_cumulative_deposit": 5250,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 16330.04
            },
            "3": {
                "period_earnings": 1006.41,
                "accumulation_period_deposit": 2625,
                "decumulation_period_deposit": 0,
```

NAV


```
                "period_withdrawal": 0,
                "cumulative_earnings": 2086.45,
                "accumulation_cumulative_deposit": 7875,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 19961.45
            },
            "4": {
                "period_earnings": 1358.01,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 11000,
                "cumulative_earnings": 3444.47,
                "accumulation_cumulative_deposit": 7875,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 11000,
                "ending_balance": 10319.47
            },
            "5": {
                "period_earnings": 832.18,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 11000,
                "cumulative_earnings": 4276.65,
                "accumulation_cumulative_deposit": 7875,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 22000,
                "ending_balance": 151.65
            }
        },
        "accumulation_balance": 19961.45
    }
```


RESPONSE

Field	Description
`on_track`	If `true`, the goal is on track.
`progress`	The goal progress percentage, defined as the current invested amount divided by the total target withdrawal amount over `d_horizon`.

NAV

Field	Description
`goal_probability`	The probability of achieving the goal with the given portfolio.
`goal_achievement_score`	A ratio of `goal_probability` to the `conf_tgt` on a scale from 0 to 100.
`action`	The recommended action. Actions represent additions to the current configuration. For example, a recommended recurring deposit of 100 indicates 100 in addition to any existing deposits.
`value`	The value of the recommended action.
`freq_unit`	The frequency unit of `value` .
`value_type`	The type of recommendation being made.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`accumulation_period_deposit`	The deposit made for this period in the accumulation phase.
`decumulation_period_deposit`	The deposit made for this period in the decumulation phase.
`period_withdrawal`	The withdrawal made for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`accumulation_cumulative_deposit`	The cumulative deposits made up to and including this period for the accumulation phase.
`decumulation_cumulative_deposit`	The cumulative deposits made up to and including this period for the decumulation phase.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.

NAV

Field	Description
`accumulation_balance`	The projected balance at the end of the accumulation horizon.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Goal`
- `Account`
- `Account Allocation`
- `Allocation`
- `Allocation Composition`
- `Model`
- `Portfolio`
- `Portfolio Asset Size`

Decumulation Goal Status

This service analyzes the state of expected goal achievement for a decumulation goal, based on the current goal configuration and context. In addition to a raw indication of whether a goal is on or off track, the service provides other metrics for advanced analysis. This framework also has built-in support for a goal recommendation engine, which recommends actions that may be taken to increase the expected goal achievement probability if a goal is off track.

HTTP REQUEST

`POST /goal_decumulation/status`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "p_ret": [0.09],
```

NAV


```
            "p_risk": [0.05],
            "trading_days_per_year": 252,
            "a_horizon": 3,
            "d_horizon": 2,
            "horizon_frequency": "year",
            "deposit_config": [
                {
                    "dep_start_reference": "a_start",
                    "dep_start_period": 0,
                    "dep_end_reference": "a_start",
                    "dep_end_period": 3,
                    "dep_amount": 2000,
                    "dep_frequency": "year",
                    "dep_inflation": 0.0
                }
            ],
            "withdrawal_config": [
                {
                    "with_amount": 11000,
                    "with_start_reference": "a_end",
                    "with_start_period": 0,
                    "with_end_reference": "d_end",
                    "with_end_period": 0,
                    "with_frequency": "year",
                    "with_inflation": 0.0
                }
            ],
            "recommendation_config": {
                "recommend": true,
                "inv_min": 0,
                "inv_max": 1000,
                "dep_min": 0,
                "dep_max": 1000,
                "horizon_min": 1,
                "horizon_max": 10,
                "recommended_inflation": 0.0
            },
            "curr_inv": 10000,
            "recommend_type": "recurring",
            "conf_tgt": 0.9,
            "n": 1000,
            "remove_outliers": true,
            "thresh_type": "perc",
            "thresh": 0,
            "withdrawal_tax": 0.0,
            "adjust_for_compounding": false,
```

NAV

```
        "compounding_rate": 0.0
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/goal_decumulation/status
```

ARGUMENTS

Parameter	Description
`p_ret` *array, required*	The annual portfolio return per period. The length of the array must be less than or equal to the sum of `a_horizon` and `d_horizon`. If it is less, values will be conformed to the number of time periods by persisting the last value. For example, [0.04, 0.06] would result in an implied `p_ret` of [0.04, 0.06, 0.06, 0.06] for a simulation that spans four time periods.
`p_risk` *array, required*	The annual portfolio standard deviation per period. The length of the array must be less than or equal to the sum of `a_horizon` and `d_horizon`. If it is less, values will be conformed to the number of time periods by persisting the last value. For example, [0.04, 0.06] would result in an implied `p_risk` of [0.04, 0.06, 0.06, 0.06] for a simulation that spans four time periods.
`curr_inv` *float, conditional requirement*	The current amount invested toward the goal.
`a_horizon` *integer, conditional requirement*	The accumulation horizon, with a frequency defined by `horizon_frequency`. For example, for a client that is looking to retire in 30 years, use `a_horizon` = 30. Must be greater than 0 if `recommend_type` is `recurring` or `combo`.
`d_horizon` *integer, conditional requirement*	The decumulation horizon, with a frequency defined by `horizon_frequency`. For example, for a client that is looking to take withdrawals for 25 years during retirement, use `d_horizon` = 25. Must be greater than 0.
`horizon_frequency` *string, conditional requirement*	The frequency in relation to the numbers defined in `a_horizon` and `d_horizon`, and at which to run the simulation. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`.
`withdrawal_config` *array(map), conditional requirement*	The withdrawals to occur during the goal's decumulation phase. `withdrawal_config` is an array of maps, with each map containing the fields shown below.

NAV

Parameter	Description
`with_amount` *float, conditional requirement*	The withdrawal amount, in today's dollars.
`with_start_reference` *string*	The reference for the starting point of the withdrawal. May be `a_end` or `d_end`, which refer to the end of the accumulation phase and the end of the decumulation phase, respectively. Defaults to `a_end`.
`with_start_period` *integer*	The starting period for the withdrawal, relative to `with_start_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `with_frequency`. Defaults to 0.
`with_end_reference` *string*	The reference for the ending point of the withdrawal. May be `a_end` or `d_end`, which refer to the end of the accumulation phase and the end of the decumulation phase, respectively. Defaults to `d_end`.
`with_end_period` *integer*	The ending period for the withdrawal, relative to `with_end_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `with_frequency`. Defaults to 0.
`with_frequency` *string*	The frequency of the withdrawal. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`. Defaults to `year`.
`with_inflation` *float*	The annualized inflation rate for the withdrawal. Defaults to 0.
`deposit_config` *array(map)*	The deposits to occur over the course of the goal horizon. `deposit_config` is an array of maps, with each map containing the fields shown below.
`dep_start_reference` *string*	The reference for the starting point of the deposit. May be `a_start`, `a_end`, or `d_end`, which refer to the start of the accumulation phase, the end of the accumulation phase, and the end of the decumulation phase, respectively. Defaults to `a_start`.
`dep_start_period` *integer*	The starting period for the deposit, relative to `dep_start_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number

NAV

Parameter	Description
	indicates a period before the reference point. Periodicity depends upon `dep_frequency`. Defaults to 0.
`dep_end_reference` *string*	The reference for the ending point of the deposit. May be `a_start`, `a_end`, or `d_end`, which refer to the start of the accumulation phase, the end of the accumulation phase, and the end of the decumulation phase, respectively. Defaults to `a_end`.
`dep_end_period` *integer*	The ending period for the deposit, relative to `dep_end_reference`. A value of 0 corresponds to the reference point, a positive number indicates a period after the reference point, and a negative number indicates a period before the reference point. Periodicity depends upon `dep_frequency`. Defaults to 0.
`dep_amount` *float*	The deposit amount, in today's dollars. Defaults to 0.
`dep_frequency` *string*	The frequency of the deposit. Must be one of the following: `year`, `quarter`, `month`, `week`, `day`. Defaults to `year`.
`dep_inflation` *float*	The annualized inflation rate for the deposit. Defaults to 0.
`recommendation_config` *map*	Information to configure the goal recommendation engine, including the fields shown below.
`recommend` *boolean*	If `true`, generate a recommended action to improve goal probability. Recommendations are only generated if a goal is off-track. Defaults to `true`.
`inv_min` *float*	The lower bound for `one-time` deposit recommendations. Defaults to 0.
`inv_max` *float*	The upper bound for `one-time` deposit recommendations. Defaults to the first `with_amount` found in `withdrawal_config` multiplied by `d_horizon`.
`dep_min` *float*	The lower bound for `recurring` deposit recommendations. Defaults to 0.
`dep_max` *float*	The upper bound for `recurring` deposit recommendations. Defaults to the first `with_amount` found in `withdrawal_config` multiplied by

NAV

Parameter	Description
	`d_horizon` divided by `a_horizon`.
`horizon_min` *integer*	The minimum number of periods (defined by `horizon_frequency`) that the accumulation horizon may be extended when `recommend_type = horizon`. Defaults to 1.
`horizon_max` *integer*	The maximum number of periods (defined by `horizon_frequency`) that the accumulation horizon may be extended when `recommend_type = horizon`. Defaults to 64.
`recommended_inflation` *float*	The annualized inflation rate attributed to a recommended recurring deposit.
`recommend_type` *string*	The type of recommended action. Value may be `recurring`, `one-time`, `combo`, or `horizon`. `recurring` returns a recurring periodic deposit. `one-time` returns a one-time deposit at the current time (occurring at time-zero, and reflected by an effective increase in the initial balance). `combo` returns a combination of recurring and one-time deposits. `horizon` returns a new accumulation horizon for the goal. Defaults to `horizon`.
`conf_tgt` *float*	The confidence target for goal achievement. `conf_tgt` indicates the level of statistical confidence that applies to the goal analysis. In some cases, the stipulated `conf_tgt` may be impossible to achieve within the given goal parameters; in these cases, the service will attempt to come as close as possible to satisfying the confidence target. Defaults to 0.90.
`n` *integer*	The number of simulations to run in the monte carlo simulation. Larger `n` values will make goal simulation results more consistent, but increase the execution time of the service. It has a maximum allowable value of 10000. Defaults to 2000.
`remove_outliers` *boolean*	If `true`, remove outlying results. If `true`, outlier analysis is performed on a median absolute deviation (MAD) basis, at the 2.5 MAD threshold. Defaults to true.
`thresh_type` *string*	The goal deviation threshold type. Value may be “perc” or “amnt”. `perc` indicates a percentage-based goal deviation threshold. `amnt` indicates a dollar-based goal deviation threshold. Defaults to `perc`.
`thresh` *float*	The goal deviation threshold value, corresponding to `thresh_type`. Defaults to 0.

NAV

Parameter	Description
`withdrawal_tax` *float*	The tax rate to apply to all withdrawals. Defaults to 0.
`trading_days_per_year` *integer*	The number of days per year for which a portfolio is subject to market fluctuation. `trading_days_per_year` impacts goal simulation results when `horizon_frequency = day`. Defaults to 252.
`adjust_for_compounding` *boolean*	If `true`, adjust periodic deposit amounts for compounding based on `compounding_rate`. This applies when a deposit's `dep_frequency` is shorter than `horizon_frequency`, as deposits are conformed to `horizon_frequency` during the analysis. Defaults to `false`.
`compounding_rate` *float*	The annualized rate to use when approximating a compounding effect on deposits. This value must be defined and `adjust_for_compounding` must be `true` in order to activate compounding adjustment. Defaults to 0.
`client_id` *UUID, conditional requirement*	The ID of a Nucleus client used to derive one or more of the following: `curr_inv`, `a_horizon`, `d_horizon`, `horizon_frequency`, and `withdrawal_config.with_amount`. If one or more of these required parameters are not provided, both `client_id` and `goal_id` are required.
`goal_id` *UUID, conditional requirement*	The ID of a Nucleus goal used to derive one or more of the following: `curr_inv`, `a_horizon`, `d_horizon`, `horizon_frequency`, and `withdrawal_config.with_amount`. If one or more of these required parameters are not provided, both `client_id` and `goal_id` are required.
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

Example Response

```
{
    "current_status": {
        "on_track": false,
```

NAV


```
        "progress": 0.4545,
        "goal_probability": 0.5298,
        "goal_achievement_score": 59,
        "return_details": {
            "0": {
                "period_earnings": 0,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 0,
                "accumulation_cumulative_deposit": 0,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 10000
            },
            "1": {
                "period_earnings": 302.11,
                "accumulation_period_deposit": 2000,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 302.11,
                "accumulation_cumulative_deposit": 2000,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 12302.11
            },
            "2": {
                "period_earnings": 722.99,
                "accumulation_period_deposit": 2000,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 1025.1,
                "accumulation_cumulative_deposit": 4000,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
                "ending_balance": 15025.1
            },
            "3": {
                "period_earnings": 971.45,
                "accumulation_period_deposit": 2000,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 1996.55,
                "accumulation_cumulative_deposit": 6000,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 0,
```

NAV ☰


```
                "ending_balance": 17996.55
            },
            "4": {
                "period_earnings": 1275.95,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 11000,
                "cumulative_earnings": 3272.5,
                "accumulation_cumulative_deposit": 6000,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 11000,
                "ending_balance": 8272.5
            },
            "5": {
                "period_earnings": 0,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 8272.5,
                "cumulative_earnings": 3272.5,
                "accumulation_cumulative_deposit": 6000,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 19272.5,
                "ending_balance": 0
            }
        },
        "accumulation_balance": 17996.55
    },
    "recommended_status": {
        "on_track": true,
        "progress": 0.4545,
        "goal_probability": 0.9043,
        "goal_achievement_score": 100,
        "action": [
            {
                "value": 593.75,
                "freq_unit": "year",
                "value_type": "recurring_deposit"
            }
        ],
        "return_details": {
            "0": {
                "period_earnings": 0,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 0,
                "cumulative_earnings": 0,
```

NAV III


```
            "accumulation_cumulative_deposit": 0,
            "decumulation_cumulative_deposit": 0,
            "cumulative_withdrawals": 0,
            "ending_balance": 10000
        },
        "1": {
            "period_earnings": 280.94,
            "accumulation_period_deposit": 2593.75,
            "decumulation_period_deposit": 0,
            "period_withdrawal": 0,
            "cumulative_earnings": 280.94,
            "accumulation_cumulative_deposit": 2593.75,
            "decumulation_cumulative_deposit": 0,
            "cumulative_withdrawals": 0,
            "ending_balance": 12874.69
        },
        "2": {
            "period_earnings": 700.92,
            "accumulation_period_deposit": 2593.75,
            "decumulation_period_deposit": 0,
            "period_withdrawal": 0,
            "cumulative_earnings": 981.86,
            "accumulation_cumulative_deposit": 5187.5,
            "decumulation_cumulative_deposit": 0,
            "cumulative_withdrawals": 0,
            "ending_balance": 16169.36
        },
        "3": {
            "period_earnings": 1125.06,
            "accumulation_period_deposit": 2593.75,
            "decumulation_period_deposit": 0,
            "period_withdrawal": 0,
            "cumulative_earnings": 2106.92,
            "accumulation_cumulative_deposit": 7781.25,
            "decumulation_cumulative_deposit": 0,
            "cumulative_withdrawals": 0,
            "ending_balance": 19888.17
        },
        "4": {
            "period_earnings": 1284.08,
            "accumulation_period_deposit": 0,
            "decumulation_period_deposit": 0,
            "period_withdrawal": 11000,
            "cumulative_earnings": 3391,
            "accumulation_cumulative_deposit": 7781.25,
            "decumulation_cumulative_deposit": 0,
```

NAV


```
                "cumulative_withdrawals": 11000,
                "ending_balance": 10172.25
            },
            "5": {
                "period_earnings": 888.86,
                "accumulation_period_deposit": 0,
                "decumulation_period_deposit": 0,
                "period_withdrawal": 11000,
                "cumulative_earnings": 4279.86,
                "accumulation_cumulative_deposit": 7781.25,
                "decumulation_cumulative_deposit": 0,
                "cumulative_withdrawals": 22000,
                "ending_balance": 61.11
            }
        },
        "accumulation_balance": 19888.17
    }
}
```


RESPONSE

Field	Description
`current_status`	The current status of the goal.
`on_track`	If `true`, the goal is on track.
`progress`	The goal progress percentage, defined as the current invested amount divided by the total target withdrawal amount over `d_horizon`.
`goal_probability`	The probability of achieving the goal with the given portfolio.
`goal_achievement_score`	A ratio of `goal_probability` to the `conf_tgt` on a scale from 0 to 100.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.

NAV

Field	Description
`accumulation_period_deposit`	The deposit made for this period in the accumulation phase.
`decumulation_period_deposit`	The deposit made for this period in the decumulation phase.
`period_withdrawal`	The withdrawal made for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`accumulation_cumulative_deposit`	The cumulative deposits made up to and including this period for the accumulation phase.
`decumulation_cumulative_deposit`	The cumulative deposits made up to and including this period for the decumulation phase.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.
`accumulation_balance`	The projected balance at the end of the accumulation horizon.
`recommended_status`	The goal status based on the provided recommendation.
`on_track`	If `true`, the goal is on track.
`progress`	The goal progress percentage, defined as the current invested amount divided by the total target withdrawal amount over `d_horizon`.
`goal_probability`	The probability of achieving the goal with the given portfolio.
`goal_achievement_score`	A ratio of `goal_probability` to the `conf_tgt` on a scale from 0 to 100.
`action`	The recommended action. Actions represent additions to the current configuration. For example, a recommended

NAV

Field	Description
	recurring deposit of 100 indicates 100 in addition to any existing deposits.
`value`	The value of the recommended action.
`freq_unit`	The frequency unit of `value` .
`value_type`	The type of recommendation being made.
`return_details`	Portfolio return information over the length of the horizon, broken down by period.
`period_earnings`	The investment earnings for this period. Earnings are calculated at the beginning of each period, before contributions or withdrawals are made.
`accumulation_period_deposit`	The deposit made for this period in the accumulation phase.
`decumulation_period_deposit`	The deposit made for this period in the decumulation phase.
`period_withdrawal`	The withdrawal made for this period.
`cumulative_earnings`	The cumulative investment earnings made up to and including this period.
`accumulation_cumulative_deposit`	The cumulative deposits made up to and including this period for the accumulation phase.
`decumulation_cumulative_deposit`	The cumulative deposits made up to and including this period for the decumulation phase.
`cumulative_withdrawals`	The cumulative withdrawals made up to and including this period.
`ending_balance`	The ending balance, inclusive of earnings and contributions for the current period.
`accumulation_balance`	The projected balance at the end of the accumulation horizon.

NAV

Field	Description
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Goal`
- `Account`
- `Account Allocation`
- `Allocation`
- `Allocation Composition`
- `Model`
- `Portfolio`
- `Portfolio Asset Size`

Performance Calculator

Performance is tracked via a series of statistics and can be tracked at the client, account, portfolio, allocation, model, goal, security, or benchmark level. Statistics are pre-defined and can be obtained in the statistics resource table.

Hydrogen uses two methods to calculate performance:

IRR (Internal Rate of Return)

IRR, also known as dollar-weighted return, calculates the performance of a portfolio while incorporating inflows and outflows. Therefore, the timing of deposits and withdrawals substantially affects the return rate, with larger cash movements having a greater impact on performance compared to smaller changes in cash. IRR is a useful tool to calculate the absolute return of a portfolio and for determining whether a portfolio is growing at a return necessary to meet an investment goal. Therefore, IRR is used mostly by Hydrogen for account and client-related performance measures.

TWR (Time Weighted Return)

TWR captures the true performance of a portfolio by removing the impact of cash inflows and outflows on the portfolio's return. It reflects the effects of portfolio composition on return, without considering the effects of the client's deposits or withdrawals. TWR captures the return of the very first investment into a portfolio. At Hydrogen, we use TWR for all performance calculations which do not have cash flows. This includes

NAV

allocation, benchmark, model, and security performance, which do not change based on total assets invested in them. TWR is also required by the Global Investment Performance Standards (GIPS) published by the CFA Institute. Calculating returns using TWR is the better method of calculating a manager's performance and analyzing the performance of a portfolio's underlying assets.

HTTP REQUEST

```
POST /performance_calculator
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "account_id": "f523d2b9-ceb8-4181-b794-40d15f720046",
        "stat_name": "ann_return"
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/performance_calculator
```

ARGUMENTS

Parameter	Description
`account_id` *UUID, conditional requirement*	The ID of a Nucleus Account to analyze. Conditionally required to provide only one of the following: `account_id`, `allocation_id`, `benchmark_id`, `client_id`, `goal_id`, `household_id`, `model_id`, `portfolio_id`, or `security_id`. For certain statistics, a `benchmark_id` may be provided in addition to any of the IDs listed.
`allocation_id` *UUID, conditional requirement*	The ID of a Nucleus Allocation to analyze. Conditionally required to provide only one of the following: `account_id`, `allocation_id`, `benchmark_id`, `client_id`, `goal_id`, `household_id`, `model_id`, `portfolio_id`, or `security_id`. For certain statistics, a `benchmark_id` may be provided in addition to any of the IDs listed.
`benchmark_id` *UUID, conditional requirement*	The ID of a Nucleus Benchmark to analyze. Conditionally required to provide only one of the following: `account_id`, `allocation_id`, `benchmark_id`, `client_id`, `goal_id`, `household_id`, `model_id`, `portfolio_id`, or `security_id`. A `benchmark_id` may be provided in addition to any of the IDs listed, but only for the

NAV

Parameter	Description
	following statistics: `beta`, `treynor_ratio`, `alpha`, `correlation`, `covariance`, `r_squared`, `information_ratio`, `tracking_error`, `active_premium`, `up_pct_ratio`, `down_pct_ratio`, `pct_gain_ratio`, `tstat`, `down_capture`, or `up_capture`.
`client_id` *UUID, conditional requirement*	The ID of a Nucleus Client to analyze. Conditionally required to provide only one of the following: `account_id`, `allocation_id`, `benchmark_id`, `client_id`, `goal_id`, `household_id`, `model_id`, `portfolio_id`, or `security_id`. For certain statistics, a `benchmark_id` may be provided in addition to any of the IDs listed.
`goal_id` *UUID, conditional requirement*	The ID of a Nucleus Goal to analyze. Conditionally required to provide only one of the following: `account_id`, `allocation_id`, `benchmark_id`, `client_id`, `goal_id`, `household_id`, `model_id`, `portfolio_id`, or `security_id`. For certain statistics, a `benchmark_id` may be provided in addition to any of the IDs listed.
`household_id` *UUID, conditional requirement*	The ID of a Nucleus Household to analyze. Conditionally required to provide only one of the following: `account_id`, `allocation_id`, `benchmark_id`, `client_id`, `goal_id`, `household_id`, `model_id`, `portfolio_id`, or `security_id`. For certain statistics, a `benchmark_id` may be provided in addition to any of the IDs listed.
`model_id` *UUID, conditional requirement*	The ID of a Nucleus Model to analyze. Conditionally required to provide only one of the following: `account_id`, `allocation_id`, `benchmark_id`, `client_id`, `goal_id`, `household_id`, `model_id`, `portfolio_id`, or `security_id`. For certain statistics, a `benchmark_id` may be provided in addition to any of the IDs listed.
`portfolio_id` *UUID, conditional requirement*	The ID of a Nucleus Portfolio to analyze. Conditionally required to provide only one of the following: `account_id`, `allocation_id`, `benchmark_id`, `client_id`, `goal_id`, `household_id`, `model_id`, `portfolio_id`, or `security_id`. For certain statistics, a `benchmark_id` may be provided in addition to any of the IDs listed.

NAV

Parameter	Description
`security_id` *UUID, conditional requirement*	The ID of a Nucleus Security to analyze. Conditionally required to provide only one of the following: `account_id`, `allocation_id`, `benchmark_id`, `client_id`, `goal_id`, `household_id`, `model_id`, `portfolio_id`, or `security_id`. For certain statistics, a `benchmark_id` may be provided in addition to any of the IDs listed.
`stat_name` *string, required*	The statistic you want to run on the Nucleus entity. See Statistics for all available options.
`start_date` *date*	Start date for the period for which to calculate the statistic.
`end_date` *date*	End date for the period for which to calculate the statistic.
`period_type` *string*	Period to carry out statistic. Options are `Y` for `annually`, `Q` for `quarterly`, `M` for `monthly`, `D` for `daily` (must use capital letters)
`n_rolling_volatility` *integer*	If `stat = rolling_n_day_vol`, number of days. Default is `7`.
`n_day_returns` *integer*	If `stat = rolling_n_day_return`, number of days. Default is `7`.
`moving_average` *integer*	If `stat = moving_avg_n_day`, number of days. Default is `7`.
`annualized_return_period` *string*	If `stat = ann_return`, period to calculate return. Default is `D`.
`active_premium_period` *string*	If `stat = active_premium`, period to calculate return. Default is `D`.
`hist_factor` *float*	If `stat = histogram`, number of data bins. Default is `5`.
`var_conf_interval` *float*	If `stat = var`, confidence interval for the var.
`n_rolling_max_drawdown` *integer*	If `stat = rolling_n_day_max_drawdown`, number of days. Default is `7`.

NAV

Parameter	Description
`risk_free_sharpe` *float*	If `stat = sharpe_ratio`, risk free rate of return. Default is `0`.
`risk_free_treynor` *float*	If `stat = treynor_ratio`, risk free rate of return. Default is `0`.
`risk_free_alpha` *float*	If `stat = alpha`, risk free rate of return. Default is `0`.
`risk_free_sortino` *float*	If `stat = sortino_ratio`, risk free rate of return. Default is `0`.
`mar_down_side_deviation` *float*	If `stat = downside_deviation`, minimum acceptable return. Default is `0`.
`num_sim_monte_carlo` *integer*	If `stat = monte_carlo`, number of Monte Carlo simulations. Default is `1000`.
`n_path_monte_carlo` *integer*	If `stat = monte_carlo`, number of Monte Carlo paths. Default is `100`.
`max_percentile_monte_carlo` *float*	If `stat = monte_carlo`, maximum probability for a Monte Carlo simulation. Default is `95`.
`min_percentile_monte_carlo` *float*	If `stat = monte_carlo`, minimum probability for a Monte Carlo simulation. Default is `5`.
`mean_percentile_monte_carlo` *float*	If `stat = monte_carlo`, mean probability for a Monte Carlo simulation. Default is `50`.

Example Response

```
{
    "ann_return": 0.06132398215253243
}
```

RESPONSE

NAV

Field	Description
`{stat_name}`	Value of the statistic for the given `stat_name`. The name of this field will always match the `stat_name` provided in the request.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Business`
- `Household`
- `Account`
- `Portfolio`
- `Portfolio Asset Size`
- `Allocation`
- `Model`
- `Model Asset Size`
- `Benchmark`
- `Goal`
- `Security`
- `Security Price`

Personal Financial Management

Budget Calculator

Budget Calculator analyzes a client's spending patterns against pre-defined budgets over time. This is useful to gain a deeper understanding of how closely a client follows a particular budget. A breakdown of each underlying budget item also provides helpful data to aid clients in staying on track and achieving their budgetary goals.

HTTP REQUEST

`POST /budget_calculator`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
```

NAV

```
    -d '{
            "budget_id": "790c936b-015d-4d8a-82ed-434a4bbc13e8",
            "as_of_date": "2016-12-22",
            "lookback_periods": 3
        }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/budget_calculator"
```

ARGUMENTS

Parameter	Description
`budget_id` *UUID, conditional requirement*	The ID of the Nucleus Budget. Please note that this service is not compatible with budgets that have a `frequency_unit` of `semi-monthly`. Required if `budget_details` is not provided.
`scope` *string*	The scope of data to be considered. Value must be one of the following: `all`, `external`, `internal`. Defaults to `all`, which will consider `internal` data, or Accounts, Portfolios, and Portfolio Transactions, along with `external` data, or Aggregation Accounts and Aggregation Account Transactions. Only applicable when neither `account_ids` or `aggregation_accounts` is populated via `budget_details` or the Nucleus Budget record indicated by `budget_id`.
`transaction_status_scope` *array[string]*	If populated, only transactions whose `status` matches one of the array values are considered in the analysis. Defaults to `null`, meaning all transactions are considered regardless of the `status`.
`currency_conversion` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Records will be converted from their original currency to the one indicated here. Conversions are supported both to and from the following currency codes: USD, GBP, EUR, AUD, CAD, CHF. See Currencies. Only applicable when a `budget_id` is provided.
`as_of_date` *date*	Reference date of the analysis. Calculations will run through the earlier of this date and `budget.end_date` as defined in Nucleus. Defaults to today's date.
`lookback_periods` *integer*	Number of lookback periods to analyze. Each period length is defined by the combination of `budget.frequency` and `budget.frequency_unit` as defined in Nucleus, and period dates

NAV

Parameter	Description
	are set on a discrete calendar basis. A value of 0 would reflect only the current (partial) period containing `as_of_date`. Defaults to 1.
`show_budget_track` *boolean*	If `true`, return the `budget_track` response, an analysis of spending versus budget for each budget period. Defaults to `true`.
`show_average_spend` *boolean*	If `true`, return the `average_spend` response, an analysis of average spending values for each budget component and in total across the time periods analyzed. Defaults to `false`.
`only_cleansed` *boolean*	If `true`, only Portfolio Transactions with the `is_cleansed` parameter set to `true` will be considered. Defaults to `false`.
`budget_details` *map, conditional requirement*	Raw budget details that can be provided in place of a `budget_id`. Allows the analysis to be run without creating a Budget in Nucleus. This field is required if `budget_id` is not provided.
`client_id` *UUID, required*	The ID of the Nucleus Client the budget belongs to.
`account_id` *UUID*	The ID of the Nucleus Account associated with the budget.
`aggregation_accounts` *array[map]*	List of Nucleus Aggregation Accounts associated with the budget.
`aggregation_account_id` *UUID, required*	The ID of the Nucleus Aggregation Account mapped to the budget.
`frequency_unit` *string, required*	Frequency of the budget. Value may be `daily`, `weekly`, `bi-weekly`, `monthly`, `quarterly`, or `annually`.
`frequency` *integer*	Number of `frequency_unit` between each budget. For example, if the `frequency_unit` is weekly and the `frequency` is 2, this means the budget occurs every two weeks. Default is 1.
`currency_code` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Only records with this currency code will be considered. Defaults to USD. See Currencies.
`total_value` *float, conditional requirement*	Total value of the budget. Required if `budget` is not provided.

NAV

Parameter	Description
`budget` *array[map], conditional requirement*	List of budget components and their values. Required if `total_value` is not provided.
`category` *string, required*	Category of the budget component.
`subcategory` *string*	Subcategory of the budget component.
`value` *float, required*	Value of the budget component.
`start_date` *date*	The start date for the budget. If not provided, defaults to today's date.
`end_date` *date*	The end date for the budget.

Example Response


```
{
    "currency_code": "USD",
    "budget_track": [
        {
            "period_start": "2016-10-01",
            "period_end": "2016-10-31",
            "total_funds_budgeted": 100.0,
            "total_funds_spent": 56.51,
            "total_funds_remaining": 43.49,
            "budget_components": [
                {
                    "category": "Food & Dining",
                    "subcategory": "Alcohol & Bars",
                    "funds_budgeted": 50.0,
                    "funds_spent": 0,
                    "funds_remaining": 50.0
                },
                {
                    "category": "Health & Fitness",
```

NAV


```
                "subcategory": "Sports",
                "funds_budgeted": 50.0,
                "funds_spent": 56.51,
                "funds_remaining": -6.51
            }
        ]
    },
    {
        "period_start": "2016-11-01",
        "period_end": "2016-11-30",
        "total_funds_budgeted": 100.0,
        "total_funds_spent": 0,
        "total_funds_remaining": 100.0,
        "budget_components": [
            {
                "category": "Food & Dining",
                "subcategory": "Alcohol & Bars",
                "funds_budgeted": 50.0,
                "funds_spent": 0,
                "funds_remaining": 50.0
            },
            {
                "category": "Health & Fitness",
                "subcategory": "Sports",
                "funds_budgeted": 50.0,
                "funds_spent": 0,
                "funds_remaining": 50.0
            }
        ]
    },
    {
        "period_start": "2016-12-01",
        "period_end": "2016-12-22",
        "total_funds_budgeted": 100.0,
        "total_funds_spent": 284.77,
        "total_funds_remaining": -184.77,
        "budget_components": [
            {
                "category": "Food & Dining",
                "subcategory": "Alcohol & Bars",
                "funds_budgeted": 50.0,
                "funds_spent": 284.77,
                "funds_remaining": -234.77
            },
            {
                "category": "Health & Fitness",
```

NAV

```
                    "subcategory": "Sports",
                    "funds_budgeted": 50.0,
                    "funds_spent": 0,
                    "funds_remaining": 50.0
                }
            ]
        }
    ]
}
```

RESPONSE

Field	Description
`currency_code`	Currency code associated with monetary response values.
`budget_track`	Analysis of spending versus budget for each budget period
`period_start`	Start date of the budget period.
`period_end`	End date of the budget period.
`total_funds_budgeted`	Total amount of funds originally budgeted.
`total_funds_spent`	Total amount of funds spent.
`total_funds_remaining`	Total amount of funds remaining, i.e. the delta between `total_funds_budgeted` and `total_funds_spent`.
`percent_spent`	Total percent of funds spent.
`percent_remaining`	Total percent of funds remaining.
`budget_components`	Details for each item defined under the budget.
`category`	The budget component's spending category.
`subcategory`	The budget component's spending subcategory.
`funds_budgeted`	Amount of funds originally budgeted.

NAV

Field	Description
`funds_spent`	Amount of funds spent.
`funds_remaining`	Amount of funds remaining, i.e. the delta between `funds_budgeted` and `funds_spent`.
`percent_spent`	Percent of funds spent.
`percent_remaining`	Percent of funds remaining.
`average_spend`	Analysis of average spending values for each budget component and in total across the time periods analyzed.
`total`	Total average amount of funds spent.
`budget_components`	Details of average spending values for each budget component.
`category`	Category of the budget component.
`subcategory`	Subcategory of the budget component.
`value`	Average amount of funds spent for the budget component.

NUCLEUS DATA DEPENDENCIES

- Client
- Account
- Portfolio
- Portfolio Transaction
- Aggregation Account
- Aggregation Account Transaction
- Budget

NAV ☰

Cash Flow Analysis

Cash Flow Analysis provides a benchmarked view of a client's cash flows over time, including income, spending, and net cash flow values. This tool is useful for charting cash flows trends over time, as well as for displaying helpful details about a client's spending habits. Tracking these cash flow values against a benchmark time period allows for insight into how the client's cash flow situation is changing relative to previous behavior.

HTTP REQUEST

Example Request

```
POST /cash_flow_analysis
```

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "client_id": "a32a48f7-d30e-489b-9d2b-576a3939343f",
        "scope": "all",
        "currency_code": "USD",
        "start_date": "2015-06-15",
        "end_date": "2015-06-17",
        "show_history": true,
        "show_spending_details": true,
        "show_income_details": true,
        "only_cleansed": true
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/cash_flow_analysis"
```

ARGUMENTS

Parameter	Description
`client_id` *UUID, conditional requirement*	The ID of a Nucleus client whose data to analyze. Conditionally required if one of `household_id`, `account_ids`, or `aggregation_account_ids` is not provided.
`household_id` *UUID, conditional requirement*	The ID of a Nucleus household whose data to analyze. Conditionally required if one of `client_id`, `account_ids`, or `aggregation_account_ids` is not provided.

NAV

Parameter	Description
`account_ids` *array[UUID], conditional requirement*	ID(s) of the Nucleus Account(s) whose data to analyze. Conditionally required if one of `client_id`, `household_id`, or `aggregation_account_ids` is not provided.
`aggregation_account_ids` *array[UUID], conditional requirement*	An array of ID(s) of Nucleus Aggregation Account(s) whose data to analyze. Conditionally required if one of `client_id`, `household_id`, or `account_ids` is not provided.
`scope` *string*	The scope of data to be considered. Value must be one of the following: `all`, `external`, `internal`. Defaults to `all`, which will consider `internal` data, or Accounts, Portfolios, and Portfolio Transactions, along with `external` data, or Aggregation Accounts and Aggregation Account Transactions. Only applicable when `client_id` or `household_id` is passed.
`transaction_status_scope` *array[string]*	If populated, only transactions whose `status` matches one of the array values are included in the analysis. Defaults to `null`, meaning all transactions are included regardless of the `status`.
`currency_code` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Only records with this currency code will be considered. Defaults to USD if `currency_conversion` is not provided. If `currency_conversion` is provided, all origin currencies will be considered by default. See Currencies.
`currency_conversion` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Records will be converted from their original currency to the one indicated here. Conversions are supported both to and from the following currency codes: USD, GBP, EUR, AUD, CAD, CHF. See Currencies.
`start_date` *date*	Start date of the analysis period. Defaults to the earliest available transaction date.
`end_date` *date*	End date of the analysis period. Defaults to the latest available transaction date.
`benchmark_start_date` *date*	Start date of the benchmark analysis period. Default values for `benchmark_start_date` and `benchmark_end_date` are based on the difference between `start_date` and `end_date`. For example, if `start_date` and `end_date` are `2016-06-15` and `2016-06-17`,

NAV

Parameter	Description
	respectively, `benchmark_start_date` and `benchmark_end_date` default to `2016-06-12` and `2016-06-14`, respectively.
`benchmark_end_date` *date*	End date of the benchmark analysis period. Default values for `benchmark_start_date` and `benchmark_end_date` are based on the difference between `start_date` and `end_date`. For example, if `start_date` and `end_date` are `2016-06-15` and `2016-06-17`, respectively, `benchmark_start_date` and `benchmark_end_date` default to `2016-06-12` and `2016-06-14`, respectively.
`show_history` *boolean*	If `true`, return a daily history of the client's cash flow details within the specified date range. Defaults to `false`.
`show_spending_details` *boolean*	If `true`, return advanced spending details including breakdowns by category and by merchant as well as any outlying expenditures. Outlier analysis is performed on a median absolute deviation (MAD) basis, at the 2.5 threshold. Defaults to `false`.
`show_income_details` *boolean*	If `true`, return advanced income details including breakdowns by category and by merchant as well as any outlying inflows. Outlier analysis is performed on a median absolute deviation (MAD) basis, at the 2.5 threshold. Defaults to `false`.
`only_cleansed` *boolean*	If `true`, only Portfolio Transactions with the `is_cleansed` parameter set to `true` will be considered. Defaults to `false`.

Example Response


```
{
    "currency_code": "USD",
    "income_summary": {
        "total": 0.0,
        "benchmark_total": 0.0,
        "change": {
            "value": 0.0,
            "percentage": null
        }
    },
    "spending_summary": {
        "total": 266.35,
```

NAV


```
            "benchmark_total": 460.96,
            "change": {
                "value": -194.61,
                "percentage": -0.4222
            }
        },
        "net_summary": {
            "total": -266.35,
            "benchmark_total": -460.96,
            "change": {
                "value": 194.61,
                "percentage": -0.4222
            }
        },
        "history": [
            {
                "date": "2015-06-15",
                "period_income": 0.0,
                "period_spending": 0.0,
                "period_net": 0.0,
                "cumulative_income": 0.0,
                "cumulative_spending": 0.0,
                "cumulative_net": 0.0
            },
            {
                "date": "2015-06-16",
                "period_income": 0.0,
                "period_spending": 0.0,
                "period_net": 0.0,
                "cumulative_income": 0.0,
                "cumulative_spending": 0.0,
                "cumulative_net": 0.0
            },
            {
                "date": "2015-06-17",
                "period_income": 0.0,
                "period_spending": 266.35,
                "period_net": -266.35,
                "cumulative_income": 0.0,
                "cumulative_spending": 266.35,
                "cumulative_net": -266.35
            }
        ],
        "benchmark_history": [
            {
                "date": "2015-06-12",
```

NAV III


```
                "period_income": 0.0,
                "period_spending": 0.0,
                "period_net": 0.0,
                "cumulative_income": 0.0,
                "cumulative_spending": 0.0,
                "cumulative_net": 0.0
            },
            {
                "date": "2015-06-13",
                "period_income": 0.0,
                "period_spending": 270.96,
                "period_net": -270.96,
                "cumulative_income": 0.0,
                "cumulative_spending": 270.96,
                "cumulative_net": -270.96
            },
            {
                "date": "2015-06-14",
                "period_income": 0.0,
                "period_spending": 190.0,
                "period_net": -190.0,
                "cumulative_income": 0.0,
                "cumulative_spending": 460.96,
                "cumulative_net": -460.96
            }
        ],
        "spending_details": {
            "by_category": [
                {
                    "category": "Food & Dining",
                    "value": 266.35,
                    "benchmark_value": 0.0,
                    "change": {
                        "value": 266.35,
                        "percentage": null
                    },
                    "weight": 1.0,
                    "benchmark_weight": 0.0,
                    "subcategories": [
                        {
                            "subcategory": "Restaurants",
                            "value": 266.35,
                            "benchmark_value": 0.0,
                            "change": {
                                "value": 266.35,
                                "percentage": null
```

NAV


```
                    },
                    "weight": 1.0,
                    "benchmark_weight": 0.0
                }
            ]
        },
        {
            "category": "Business Services",
            "value": 0.0,
            "benchmark_value": 270.96,
            "change": {
                "value": -270.96,
                "percentage": -1.0
            },
            "weight": 0.0,
            "benchmark_weight": 0.5878,
            "subcategories": [
                {
                    "subcategory": "Legal",
                    "value": 0.0,
                    "benchmark_value": 270.96,
                    "change": {
                        "value": -270.96,
                        "percentage": -1.0
                    },
                    "weight": 0.0,
                    "benchmark_weight": 1.0
                }
            ]
        },
        {
            "category": "Health & Fitness",
            "value": 0.0,
            "benchmark_value": 190.0,
            "change": {
                "value": -190.0,
                "percentage": -1.0
            },
            "weight": 0.0,
            "benchmark_weight": 0.4122,
            "subcategories": [
                {
                    "subcategory": "Health Insurance",
                    "value": 0.0,
                    "benchmark_value": 190.0,
                    "change": {
```

NAV


```
                        "value": -190.0,
                        "percentage": -1.0
                    },
                    "weight": 0.0,
                    "benchmark_weight": 1.0
                }
            ]
        }
    ],
    "by_merchant": [
        {
            "merchant": "Isdom",
            "value": 266.35,
            "benchmark_value": 0.0,
            "change": {
                "value": 266.35,
                "percentage": null
            },
            "weight": 1.0,
            "benchmark_weight": 0.0
        },
        {
            "merchant": "Blackzim",
            "value": 0.0,
            "benchmark_value": 270.96,
            "change": {
                "value": -270.96,
                "percentage": -1.0
            },
            "weight": 0.0,
            "benchmark_weight": 0.5878
        },
        {
            "merchant": "Geojaycare",
            "value": 0.0,
            "benchmark_value": 190.0,
            "change": {
                "value": -190.0,
                "percentage": -1.0
            },
            "weight": 0.0,
            "benchmark_weight": 0.4122
        }
    ],
    "outliers": []
},
```

NAV

```
        "income_details": {
            "by_category": [],
            "by_merchant": [],
            "outliers": []
        }
    }
```

RESPONSE

Field	Description
`currency_code`	Currency code associated with monetary response values.
`income_summary`	List of the total income values and the calculated delta between overall and benchmark.
`total`	Total cash inflows over the trend period.
`benchmark_total`	Total cash inflows over the benchmark trend period.
`change`	Difference in total cash inflows between the trend period and the benchmark trend period.
`value`	Value change as of the analysis end date.
`percentage`	Percentage change as of the analysis end date.
`spending_summary`	List of the total spending values and the calculated delta between overall and benchmark.
`total`	Total cash outflows over the trend period.
`benchmark_total`	Total cash outflows over the benchmark trend period.
`change`	Difference in total cash outflows between the trend period and the benchmark trend period.
`value`	Value change as of the analysis end date.
`percentage`	Percentage change as of the analysis end date.
`net_summary`	List of the total net values and the calculated delta between overall and benchmark.

NAV

Field	Description
`total`	Total net cash flows over the trend period.
`benchmark_total`	Total net cash flows over the benchmark trend period.
`change`	Difference in total net cash flows between the trend period and the benchmark trend period.
`value`	Value change as of the analysis end date.
`percentage`	Percentage change as of the analysis end date.
`history`	List of income, spending and net values calculated by day.
`date`	Date of the spending history record.
`period_income`	Cash inflows during the period.
`period_spending`	Cash outflows during the period.
`period_net`	Net cash inflows (outflows) during the period.
`cumulative_income`	Cumulative cash inflows up to and including this period.
`cumulative_spending`	Cumulative cash outflows up to and including this period.
`cumulative_net`	Cumulative net cash inflows (outflows) up to and including this period.
`benchmark_history`	List of benchmark income, spending and net values calculated by day.
`date`	Date of benchmark spending history record.
`period_income`	Cash inflows during the period.
`period_spending`	Cash outflows during the period.
`period_net`	Net cash inflows (outflows) during the period.
`cumulative_income`	Cumulative cash inflows up to and including this period.
`cumulative_spending`	Cumulative cash outflows up to and including this period.
`cumulative_net`	Cumulative net cash inflows (outflows) up to and including this period.

NAV

Field	Description
spending_details	List of spending information separated by categories and merchants.
by_category	List of spending information separated by categories and their relative subcategories.
category	Spending category as defined in the Nucleus transactions.
value	Sum of all transactions over the period for the given category.
benchmark_value	Sum of all transactions over the benchmark period for the given category.
change	Difference in total net cash outflows per category between the trend period and the benchmark trend period.
value	Value change as of the analysis end date.
percentage	Percentage change as of the analysis end date.
weight	The proportion of all spending over the period related to this category.
benchmark_weight	The proportion of all spending over the benchmark period related to this category.
subcategories	List of spending subcategory as defined in the Nucleus transactions.
subcategory	Spending category as defined in the Nucleus transactions.
value	Sum of all transactions over the period for the given subcategory.
benchmark_value	Sum of all transactions over the benchmark period for the given subcategory.
change	Difference in total net cash outflows per subcategory between the trend period and the benchmark trend period.
value	Value change as of the analysis end date.
percentage	Percentage change as of the analysis end date.
weight	The proportion of all spending over the period related to this subcategory.

NAV

Field	Description
benchmark_weight	The proportion of all spending over the benchmark period related to this subcategory.
by_merchant	List of spending information separated by merchant.
merchant	Merchant name as defined in the Nucleus transactions.
value	Sum of all transactions over the period for the given merchant.
benchmark_value	Sum of all transactions over the benchmark period for the given merchant.
change	Difference in total net cash outflows per merchant between the trend period and the benchmark trend period.
value	Value change as of the analysis end date.
percentage	Percentage change as of the analysis end date.
weight	The proportion of all spending over the period related to this merchant.
benchmark_weight	The proportion of all spending over the benchmark period related to this merchant.
outliers	List of spending transactions whose amount is beyond 2.5 median absolute deviations from the median of all transaction amounts. Each entry represent a raw transaction record from Nucleus. Please see Nucleus Portfolio Transaction and Aggregation Account Transaction for more details on underlying fields.
income_details	List of income information separated by categories and merchants.
by_category	List of income information separated by categories and their relative subcategories.
category	Income category as defined in the Nucleus transactions.
value	Sum of all transactions over the period for the given category.
benchmark_value	Sum of all transactions over the benchmark period for the given category.

NAV

Field	Description
change	Difference in total net cash inflows per category between the trend period and the benchmark trend period.
value	Value change as of the analysis end date.
percentage	Percentage change as of the analysis end date.
weight	The proportion of all income over the period related to this category.
benchmark_weight	The proportion of all income over the benchmark period related to this category.
subcategories	List of income subcategory as defined in the Nucleus transactions.
subcategory	Income category as defined in the Nucleus transactions.
value	Sum of all transactions over the period for the given subcategory.
benchmark_value	Sum of all transactions over the benchmark period for the given subcategory.
change	Difference in total net cash inflows per subcategory between the trend period and the benchmark trend period.
value	Value change as of the analysis end date.
percentage	Percentage change as of the analysis end date.
weight	The proportion of all income over the period related to this subcategory.
benchmark_weight	The proportion of all income over the benchmark period related to this subcategory.
by_merchant	List of income information separated by merchant.
merchant	Merchant name as defined in the Nucleus transactions.
value	Sum of all transactions over the period for the given merchant.
benchmark_value	Sum of all transactions over the benchmark period for the given merchant.

NAV

Field	Description
change	Difference in total net cash inflows per merchant between the trend period and the benchmark trend period.
value	Value change as of the analysis end date.
percentage	Percentage change as of the analysis end date.
weight	The proportion of all income over the period related to this merchant.
benchmark_weight	The proportion of all income over the benchmark period related to this merchant.
outliers	List of income transactions whose amount is beyond 2.5 median absolute deviations from the median of all transaction amounts. Each entry represent a raw transaction record from Nucleus. Please see Nucleus Portfolio Transaction and Aggregation Account Transaction for more details on underlying fields.

NUCLEUS DATA DEPENDENCIES

- Client
- Account
- Portfolio
- Portfolio Transaction
- Aggregation Account
- Aggregation Account Transaction
- Transaction Categories

Fee Analysis

Fee Analysis provides a detailed breakdown of the various fees paid across a client's accounts, including investment-related fees and banking fees. This offers insight into the size, frequency, and source of fee activity, as well as an opportunity to identify fee outlays that are unknown or undesirable. The analysis operates on transaction data stored for a given client.

HTTP REQUEST

NAV

```
POST /fee_analysis
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "client_id": "c25d94fb-ce5f-49e7-8e43-9ffd44eb7b0b",
        "start_date": "2016-01-01",
        "end_date": "2020-09-01",
        "currency_code": "USD"
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/fee_analysis"
```

ARGUMENTS

Parameter	Description
`client_id` *UUID, conditional requirement*	The ID of a Nucleus Client. Conditionally required if `aggregation_account_ids` is not passed in the request.
`aggregation_account_ids` *array[UUID], conditional requirement*	An array of ID(s) of Nucleus Aggregation Accounts. Conditionally required if `client_id` is not passed in the request.
`transaction_status_scope` *array[string]*	If populated, only transactions whose `status` matches one of the array values are considered in the analysis. Defaults to `null`, meaning all transactions are considered regardless of the `status`.
`currency_code` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Only records with this currency code will be considered. Defaults to USD if `currency_conversion` is not provided. If `currency_conversion` is provided, all origin currencies will be considered by default. See Currencies.
`currency_conversion` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Records will be converted from their original currency to the one indicated here. Conversions are supported both to and from the following currency codes: USD, GBP, EUR, AUD, CAD, CHF. See Currencies.
`start_date` *date*	Start date of the fee analysis. Defaults to earliest date there is data.

NAV

Parameter	Description
`end_date` *date*	End date of the fee analysis. Defaults to latest date there is data.

Example Response


```
{
    "analysis_start": "2016-01-01",
    "analysis_end": "2020-09-01",
    "currency_code": "USD",
    "summary": {
        "number_of_accounts": 2,
        "holding_fees": {
            "number_of_holdings": 1,
            "holding_amount": 7935.25,
            "annual_fee": 238.06,
            "average_expense_ratio": 0.03,
            "by_holding": [
                {
                    "id": "be372da6-0ccb-489e-972c-b1be44dfe0d8",
                    "ticker": "VNQ",
                    "ticker_name": "Vanguard Real Estate ETF",
                    "shares": 100.0,
                    "price": 79.35,
                    "amount": 7935.25,
                    "expense_ratio": 0.03,
                    "annual_fee": 238.06
                }
            ]
        },
        "transaction_fees": {
            "number_of_transactions": 8,
            "latest_transaction_date": "2020-06-23T00:00:00.000+0000",
            "total": 104.33,
            "min": 3.5,
            "max": 35.0,
            "median": 11.2,
            "mean": 13.04,
            "by_category": [
                {
                    "category": "Investment",
```

NAV ☰


```
            "number_of_transactions": 5,
            "latest_transaction_date": "2020-06-23T00:00:00.000+0000",
            "total": 60.83,
            "min": 7.81,
            "max": 15.76,
            "median": 13.45,
            "mean": 12.17,
            "subcategories": [
                {
                    "subcategory": "Margin Expense",
                    "number_of_transactions": 1,
                    "latest_transaction_date": "2019-08-02T00:00:00.000+0000",
                    "total": 13.45,
                    "min": 13.45,
                    "max": 13.45,
                    "median": 13.45,
                    "mean": 13.45
                },
                {
                    "subcategory": "Trade Commissions",
                    "number_of_transactions": 3,
                    "latest_transaction_date": "2020-06-03T00:00:00.000+0000",
                    "total": 39.57,
                    "min": 8.95,
                    "max": 15.76,
                    "median": 14.86,
                    "mean": 13.19
                },
                {
                    "subcategory": "Administrative Fee",
                    "number_of_transactions": 1,
                    "latest_transaction_date": "2020-06-23T00:00:00.000+0000",
                    "total": 7.81,
                    "min": 7.81,
                    "max": 7.81,
                    "median": 7.81,
                    "mean": 7.81
                }
            ]
        },
        {
            "category": "Banking & Credit",
            "number_of_transactions": 3,
            "latest_transaction_date": "2020-04-16T00:00:00.000+0000",
            "total": 43.5,
            "min": 3.5,
```

NAV ☰


```
                    "max": 35.0,
                    "median": 5.0,
                    "mean": 14.5,
                    "subcategories": [
                        {
                            "subcategory": "Banking Fee",
                            "number_of_transactions": 1,
                            "latest_transaction_date": "2020-02-01T00:00:00.000+0000",
                            "total": 5.0,
                            "min": 5.0,
                            "max": 5.0,
                            "median": 5.0,
                            "mean": 5.0
                        },
                        {
                            "subcategory": "Transfer Fee",
                            "number_of_transactions": 1,
                            "latest_transaction_date": "2020-03-21T00:00:00.000+0000",
                            "total": 3.5,
                            "min": 3.5,
                            "max": 3.5,
                            "median": 3.5,
                            "mean": 3.5
                        },
                        {
                            "subcategory": "Overdraft Fee",
                            "number_of_transactions": 1,
                            "latest_transaction_date": "2020-04-16T00:00:00.000+0000",
                            "total": 35.0,
                            "min": 35.0,
                            "max": 35.0,
                            "median": 35.0,
                            "mean": 35.0
                        }
                    ]
                }
            ]
        }
    },
    "by_account": [
        {
            "aggregation_account_id": "481cfa4e-fed0-453d-b0e0-b987aaaad459",
            "account_name": "Bank Gold Investment",
            "institution_name": "Citywide Bank",
            "holding_fees": {
                "number_of_holdings": 1,
```

NAV III


```
            "holding_amount": 7935.25,
            "annual_fee": 238.06,
            "average_expense_ratio": 0.03,
            "by_holding": [
                {
                    "id": "be372da6-0ccb-489e-972c-b1be44dfe0d8",
                    "ticker": "VNQ",
                    "ticker_name": "Vanguard Real Estate ETF",
                    "shares": 100.0,
                    "price": 79.35,
                    "amount": 7935.25,
                    "expense_ratio": 0.03,
                    "annual_fee": 238.06
                }
            ]
        },
        "transaction_fees": {
            "number_of_transactions": 5,
            "latest_transaction_date": "2020-06-23T00:00:00.000+0000",
            "total": 60.83,
            "min": 7.81,
            "max": 15.76,
            "median": 13.45,
            "mean": 12.17,
            "by_category": [
                {
                    "category": "Investment",
                    "number_of_transactions": 5,
                    "latest_transaction_date": "2020-06-23T00:00:00.000+0000",
                    "total": 60.83,
                    "min": 7.81,
                    "max": 15.76,
                    "median": 13.45,
                    "mean": 12.17,
                    "subcategories": [
                        {
                            "subcategory": "Margin Expense",
                            "number_of_transactions": 1,
                            "latest_transaction_date": "2019-08-02T00:00:00.000+0(
                            "total": 13.45,
                            "min": 13.45,
                            "max": 13.45,
                            "median": 13.45,
                            "mean": 13.45
                        },
                        {
```

NAV ☰


```
                                "subcategory": "Trade Commissions",
                                "number_of_transactions": 3,
                                "latest_transaction_date": "2020-06-03T00:00:00.000+0(
                                "total": 39.57,
                                "min": 8.95,
                                "max": 15.76,
                                "median": 14.86,
                                "mean": 13.19
                            },
                            {
                                "subcategory": "Administrative Fee",
                                "number_of_transactions": 1,
                                "latest_transaction_date": "2020-06-23T00:00:00.000+0(
                                "total": 7.81,
                                "min": 7.81,
                                "max": 7.81,
                                "median": 7.81,
                                "mean": 7.81
                            }
                        ]
                    }
                ]
            }
        },
        {
            "aggregation_account_id": "ac5f3095-bcee-431f-b200-53bc7448d259",
            "account_name": "Bank Gold Checking",
            "institution_name": "Citywide Bank",
            "holding_fees": {
                "number_of_holdings": 0,
                "holding_amount": 0,
                "annual_fee": 0,
                "average_expense_ratio": 0,
                "by_holding": []
            },
            "transaction_fees": {
                "number_of_transactions": 3,
                "latest_transaction_date": "2020-04-16T00:00:00.000+0000",
                "total": 43.5,
                "min": 3.5,
                "max": 35.0,
                "median": 5.0,
                "mean": 14.5,
                "by_category": [
                    {
                        "category": "Banking & Credit",
```

NAV


```
                    "number_of_transactions": 3,
                    "latest_transaction_date": "2020-04-16T00:00:00.000+0000",
                    "total": 43.5,
                    "min": 3.5,
                    "max": 35.0,
                    "median": 5.0,
                    "mean": 14.5,
                    "subcategories": [
                        {
                            "subcategory": "Banking Fee",
                            "number_of_transactions": 1,
                            "latest_transaction_date": "2020-02-01T00:00:00.000+0(
                            "total": 5.0,
                            "min": 5.0,
                            "max": 5.0,
                            "median": 5.0,
                            "mean": 5.0
                        },
                        {
                            "subcategory": "Transfer Fee",
                            "number_of_transactions": 1,
                            "latest_transaction_date": "2020-03-21T00:00:00.000+0(
                            "total": 3.5,
                            "min": 3.5,
                            "max": 3.5,
                            "median": 3.5,
                            "mean": 3.5
                        },
                        {
                            "subcategory": "Overdraft Fee",
                            "number_of_transactions": 1,
                            "latest_transaction_date": "2020-04-16T00:00:00.000+0(
                            "total": 35.0,
                            "min": 35.0,
                            "max": 35.0,
                            "median": 35.0,
                            "mean": 35.0
                        }
                    ]
                }
            ]
        }
    }
```

NAV ☰

```
        ]
    }
```

RESPONSE

Field	Description
`analysis_start`	Start date of the analysis.
`analysis_end`	End date of the analysis.
`currency_code`	Currency code associated with monetary response values.
`summary`	Summary of fees incurred across all aggregation accounts.
`number_of_accounts`	Number of aggregation accounts that incur fees.
`holding_fees`	Holding fees incurred on Nucleus Aggregation Account Holdings.
`number_of_holdings`	Number of holdings that incur fees.
`holding_amount`	Total amount of holdings that incur fees.
`annual_fee`	Total estimated annual holding fees to be incurred.
`average_expense_ratio`	Average expense ratio of the holdings.
`by_holding`	Holding fees by individual holding.
`id`	ID of the Nucleus Aggregation Account Holding.
`ticker`	Ticker of the holding.

NAV

Field	Description
`ticker_name`	Ticker name of the holding.
`shares`	Number of shares of the holding.
`price`	Price per share of the holding.
`amount`	Amount of the holding.
`expense_ratio`	Expense ratio of the holding.
`annual_fee`	Estimated annual holding fee to be incurred.
`transaction_fees`	Transaction fees incurred as Nucleus Aggregation Account Transactions.
`number_of_transactions`	Total number of transaction fees incurred.
`latest_transaction_date`	Latest date a transaction fee was incurred.
`total`	Total amount of fees incurred.
`min`	Minimum fee incurred.
`max`	Maximum fee incurred.
`median`	Median fee incurred.
`mean`	Average fee incurred.

NAV

Field	Description
`by_category`	Transaction fees by category and subcategory.
`category`	Category of the transactions. Value will be either `Banking & Credit` or `Investment`, depending on whether the fee was charged to a non-investment or investment aggregation account, respectively.
`number_of_transactions`	Number of transaction fees with this category.
`latest_transaction_date`	Latest date a transaction fee was incurred with this category.
`total`	Total amount of fees incurred with this category.
`min`	Minimum fee incurred with this category.
`max`	Maximum fee incurred with this category.
`median`	Median fee incurred with this category.
`mean`	Average fee incurred with this category.
`subcategories`	Transaction fees by subcategory.
`subcategory`	Subcategory of the transactions. For `Banking & Credit` category, the subcategory will reflect the `bank_credit.subcategory` value of the underlying transaction. For `Investment` category, the subcategory will reflect the `investment.investment_type` value, or will be set to `Trade Commissions` for fees charged as part of trade-related transactions.
`number_of_transactions`	Number of transaction fees with this subcategory.

NAV

Field	Description
`latest_transaction_date`	Latest date a transaction fee was incurred with this subcategory.
`total`	Total amount of fees incurred with this subcategory.
`min`	Minimum fee incurred with this subcategory.
`max`	Maximum fee incurred with this subcategory.
`median`	Median fee incurred with this subcategory.
`mean`	Average fee incurred with this subcategory.
`by_account`	Fees broken down by aggregation account.
`aggregation_account_id`	ID of the Nucleus Aggregation Account.
`account_name`	Name of the aggregation account.
`institution_name`	Name of the institution for the aggregation account.
`holding_fees`	Holding fees incurred on Nucleus Aggregation Account Holdings.
`number_of_holdings`	Number of holdings that incur fees.
`holding_amount`	Total amount of holdings that incur fees.
`annual_fee`	Total estimated annual holding fees to be incurred.
`average_expense_ratio`	Average expense ratio of the holdings.

NAV

Field	Description
`by_holding`	Holding fees by individual holding.
`id`	ID of the Nucleus Aggregation Account Holding.
`ticker`	Ticker of the holding.
`ticker_name`	Ticker name of the holding.
`shares`	Number of shares of the holding.
`price`	Price per share of the holding.
`amount`	Amount of the holding.
`expense_ratio`	Expense ratio of the holding.
`annual_fee`	Estimated annual holding fee to be incurred.
`transaction_fees`	Transaction fees incurred as Nucleus Aggregation Account Transactions.
`number_of_transactions`	Total number of transaction fees incurred.
`latest_transaction_date`	Latest date a transaction fee was incurred.
`total`	Total amount of fees incurred.
`min`	Minimum fee incurred.

NAV

Field	Description
`max`	Maximum fee incurred.
`median`	Median fee incurred.
`mean`	Average fee incurred.
`by_category`	Transaction fees by category and subcategory.
`category`	Category of the transactions. Value will be either `Banking & Credit` or `Investment`, depending on whether the fee was charged to a non-investment or investment aggregation account, respectively.
`number_of_transactions`	Number of transaction fees with this category.
`latest_transaction_date`	Latest date a transaction fee was incurred with this category.
`total`	Total amount of fees incurred with this category.
`min`	Minimum fee incurred with this category.
`max`	Maximum fee incurred with this category.
`median`	Median fee incurred with this category.
`mean`	Average fee incurred with this category.
`subcategories`	Transaction fees by subcategory.
`subcategory`	Subcategory of the transactions. For `Banking & Credit` category, the subcategory will reflect the

NAV

Field	
	`bank_credit.subcategory` value of the underlying transaction. For `Investment` category, the subcategory will reflect the `investment.investment_type` value, or will be set to `Trade Commissions` for fees charged as part of trade-related transactions.
`number_of_transactions`	Number of transaction fees with this subcategory.
`latest_transaction_date`	Latest date a transaction fee was incurred with this subcategory.
`total`	Total amount of fees incurred with this subcategory.
`min`	Minimum fee incurred with this subcategory.
`max`	Maximum fee incurred with this subcategory.
`median`	Median fee incurred with this subcategory.
`mean`	Average fee incurred with this subcategory.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Aggregation Account`
- `Aggregation Account Transaction`
- `Aggregation Account Holding`
- `Security`
- `Security Price`

Financial Health Check (Personal)

NAV

It is important to assess the health of a client's financial situation in order to provide them with recommendations and guidance. This tool provides a financial health check by generating a series of financial ratios. After collecting information about the user's assets, liabilities, income, and expenses, this tool returns ratio results and indicates a pass or fail based on a rule of thumb measure. The rule of thumb can be the system default or be overridden.

HTTP REQUEST

`POST /financial_health_check`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "liquid_assets": 5000,
        "non_liquid_assets": 10000,
        "short_term_liabilities": 11000,
        "total_liabilities": 14000,
        "gross_annual_income": 60000,
        "net_monthly_income": 3500,
        "monthly_expenses": 3000,
        "ratio_targets": {
            "liquidity_ratio_expenses": 2.5,
            "liquidity_ratio_liabilities": 0.1,
            "current_ratio": 0.5,
            "asset_allocation_ratio": 1.5,
            "savings_ratio_gross": 0.1,
            "savings_ratio_net": 0.1
        }
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/financial_health_check"
```

ARGUMENTS

Parameter	Description
`liquid_assets` *float*	Value for the sum of cash and cash equivalents, checking accounts, savings accounts, money market accounts, money market mutual funds, CDs with short maturities, and similar assets.

NAV

Parameter	Description
`non_liquid_assets` *float*	Value for non-money market investment accounts, vehicles, real estate, business interests, personal property, the cash value of insurance policies, and similar assets.
`short_term_liabilities` *float*	Value for all debt and credit obligations, charges, bills, and payments due within one year.
`total_liabilities` *float*	Value for total liabilities, examples include mortgages, car loans, credit card balances, and student loans. Includes `short_term_liabilities`.
`gross_annual_income` *float*	Total amount of income earned annually before taxes.
`net_monthly_income` *float*	Take-home monthly pay after taxes and other payroll deductions.
`monthly_expenses` *float*	Value for average monthly living expenses.
`ratio_targets` *map*	Target values for each of the available financial ratios. These values are used as a benchmark for the financial health check and determine pass or fail. More information for these ratios can be found in the `Appendix`.
`liquidity_ratio_expenses` *float*	Target ratio value for `liquidity_ratio_expenses`. Defaults to `2.5`. The ratio value is calculated as `liquid_assets` / `monthly_expenses`.
`liquidity_ratio_liabilities` *float*	Target ratio value for `liquidity_ratio_liabilities`. Defaults to `0.1`. The ratio value is calculated as `liquid_assets` / `total_liabilities`.
`current_ratio` *float*	Target ratio value for `current_ratio`. Defaults to `0.5`. The ratio value is calculated as `liquid_assets` / `short_term_liabilities`.
`asset_allocation_ratio` *float*	Target ratio value for `asset_allocation_ratio`. Defaults to `1.5`. The ratio value is calculated as `liquid_assets` / `net_worth`.

NAV

Parameter	Description
`savings_ratio_gross` *float*	Target ratio value for `savings_ratio_gross`. Defaults to `0.1`. The ratio value is calculated as `monthly_surplus` / `gross_monthly_income`.
`savings_ratio_net` *float*	Target ratio value for `savings_ratio_net`. Defaults to `0.1`. The ratio value is calculated as `monthly_surplus` / `net_monthly_income`.
`client_id` *UUID*	The ID of a Nucleus client used to derive one or more of the following based on the client's financials: `liquid_assets`, `non_liquid_assets`, `short_term_liabilities`, `total_liabilities`, `net_monthly_income`, and `monthly_expenses`. If raw values for any of these optional parameters are not provided, we will use `client_id` to try to derive those values instead.
`lookback_periods` *integer*	Number of monthly periods to analyze when deriving income and expense values using a `client_id`. Values are averaged across all periods. Defaults to 3.

Example Response


```
{
    "liquidity_ratio_expenses": {
        "ratio_result": 1.6667,
        "target_value": 2.5,
        "pass": false,
        "percentile_grade": 66
    },
    "liquidity_ratio_liabilities": {
        "ratio_result": 0.3571,
        "target_value": 0.1,
        "pass": true,
        "percentile_grade": 100
    },
    "current_ratio": {
        "ratio_result": 0.4545,
        "target_value": 0.5,
        "pass": false,
        "percentile_grade": 90
```

NAV


```
    },
    "asset_allocation_ratio": {
        "ratio_result": 5,
        "target_value": 1.5,
        "pass": true,
        "percentile_grade": 100
    },
    "savings_ratio_gross": {
        "ratio_result": 0.1,
        "target_value": 0.1,
        "pass": true,
        "percentile_grade": 100
    },
    "savings_ratio_net": {
        "ratio_result": 0.1429,
        "target_value": 0.1,
        "pass": true,
        "percentile_grade": 100
    },
    "total_assets": 15000,
    "net_worth": 1000,
    "gross_monthly_income": 5000,
    "monthly_surplus": 500
}
```


RESPONSE

Field	Description
`liquidity_ratio_expenses`	Ratio measuring available liquidity with respect to ongoing expenses.
`ratio_result`	The ratio value, calculated as `liquid_assets` / `monthly_expenses` .
`target_value`	The target ratio value, as determined by `ratio_targets` .
`pass`	A boolean indicating if `ratio_result` is sufficiently high. If `ratio_result` is greater than or equal to the corresponding `target_value` , then `pass` is `true` .
`percentile_grade`	A linear percentile score for `ratio_result` on a scale from 0 to 100, where 100 represents the corresponding `target_value` .

NAV

Field	Description
`liquidity_ratio_liabilities`	Ratio measuring available liquidity with respect to liabilities.
`ratio_result`	The ratio value, calculated as `liquid_assets` / `total_liabilities`.
`target_value`	The target ratio value, as determined by `ratio_targets`.
`pass`	A boolean indicating if `ratio_result` is sufficiently high. If `ratio_result` is greater than or equal to the corresponding `target_value`, then `pass` is `true`.
`percentile_grade`	A linear percentile score for `ratio_result` on a scale from 0 to 100, where 100 represents the corresponding `target_value`.
`current_ratio`	Ratio measuring available liquidity with respect to short-term liabilities.
`ratio_result`	The ratio value, calculated as `liquid_assets` / `short_term_liabilities`.
`target_value`	The target ratio value, as determined by `ratio_targets`.
`pass`	A boolean indicating if `ratio_result` is sufficiently high. If `ratio_result` is greater than or equal to the corresponding `target_value`, then `pass` is `true`.
`percentile_grade`	A linear percentile score for `ratio_result` on a scale from 0 to 100, where 100 represents the corresponding `target_value`.
`asset_allocation_ratio`	Ratio measuring available liquidity with respect to net worth.
`ratio_result`	The ratio value, calculated as `liquid_assets` / `net_worth`.
`target_value`	The target ratio value, as determined by `ratio_targets`.
`pass`	A boolean indicating if `ratio_result` is sufficiently high. If `ratio_result` is greater than or equal to the corresponding `target_value`, then `pass` is `true`.

NAV

Field	Description
`percentile_grade`	A linear percentile score for `ratio_result` on a scale from 0 to 100, where 100 represents the corresponding `target_value`.
`savings_ratio_gross`	Ratio measuring savings potential with respect to gross income.
`ratio_result`	The ratio value, calculated as `monthly_surplus` / `gross_monthly_income`.
`target_value`	The target ratio value, as determined by `ratio_targets`.
`pass`	A boolean indicating if `ratio_result` is sufficiently high. If `ratio_result` is greater than or equal to the corresponding `target_value`, then `pass` is `true`.
`percentile_grade`	A linear percentile score for `ratio_result` on a scale from 0 to 100, where 100 represents the corresponding `target_value`.
`savings_ratio_net`	Ratio measuring savings potential with respect to net income.
`ratio_result`	The ratio value, calculated as `monthly_surplus` / `net_monthly_income`.
`target_value`	The target ratio value, as determined by `ratio_targets`.
`pass`	A boolean indicating if `ratio_result` is sufficiently high. If `ratio_result` is greater than or equal to the corresponding `target_value`, then `pass` is `true`.
`percentile_grade`	A linear percentile score for `ratio_result` on a scale from 0 to 100, where 100 represents the corresponding `target_value`.
`total_assets`	Total assets, calculated as `liquid_assets + non_liquid_assets`.
`net_worth`	Net worth, calculated as `total_assets - total_liabilities`.
`gross_monthly_income`	Gross monthly income, calculated as `gross_annual_income` / 12.
`monthly_surplus`	Net monthly surplus, calculated as `net_monthly_income - monthly_expenses`.

NAV

NUCLEUS DATA DEPENDENCIES

- Client
- Aggregation Account
- Aggregation Account Balance
- Aggregation Account Transaction
- Account Categories
- Transaction Categories

Financial Picture

Financial Picture considers all of a client's aggregation account information in order to provide a complete overview of the client's financial situation in a specified currency and over a specified date range. For all requests, the `snapshot` response provides the client's total assets, total liabilities, and net worth, along with a categorical breakdown of where the client's wealth is positioned. The optional `history` and `change` responses provide a more detailed insight into how the client's financial picture has changed over time.

HTTP REQUEST

```
POST /financial_picture
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "client_id": "a32a48f7-d30e-489b-9d2b-576a3939343f",
        "currency_code": "USD",
        "start_date": "2015-06-15",
        "end_date": "2015-06-16",
        "show_history": true,
        "show_change": true,
        "show_category_breakdown": true,
        "create_log": true
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/financial_picture"
```

ARGUMENTS

NAV

Parameter	Description
`client_id` *UUID, conditional requirement*	The ID of a Nucleus client whose data to analyze. Conditionally required if `household_id` is not provided.
`household_id` *UUID, conditional requirement*	The ID of a Nucleus household whose data to analyze. Conditionally required if `client_id` is not provided.
`currency_code` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Only records with this currency code will be considered. Defaults to USD if `currency_conversion` is not provided. If `currency_conversion` is provided, all origin currencies will be considered by default. See Currencies.
`currency_conversion` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Records will be converted from their original currency to the one indicated here. Conversions are supported both to and from the following currency codes: USD, GBP, EUR, AUD, CAD, CHF. See Currencies.
`start_date` *date*	Start date of the financial picture analysis. Defaults to earliest date there is data.
`end_date` *date*	End date of the financial picture analysis. Defaults to latest date there is data.
`show_change` *boolean*	If `true`, return cumulative changes in the client's total assets, total liabilities, and net worth over time within the specified date range. Defaults to `false`.
`show_history` *boolean*	If `true`, return a daily history of the client's financial picture within the specified date range. Defaults to `false`.
`show_category_breakdown` *boolean*	If `true`, return detailed breakdowns by category and subcategory within the `snapshot` and `history` objects. Defaults to `true`.
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Nucleus as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

Example Response

NAV


```
{
    "currency_code": "USD",
    "snapshot": {
        "total_assets": {
            "balance": 172860.93,
            "latest_balance_time_stamp": "2015-06-16T00:00:00.000Z"
        },
        "total_liabilities": {
            "balance": 56875.25,
            "latest_balance_time_stamp": "2015-06-16T00:00:00.000Z"
        },
        "net_worth": {
            "balance": 115985.68,
            "latest_balance_time_stamp": "2015-06-16T00:00:00.000Z"
        },
        "investable_cash": {
            "balance": 90818.39,
            "latest_balance_time_stamp": "2015-06-16T00:00:00.000Z"
        },
        "retirement_savings": {
            "balance": 0.0,
            "latest_balance_time_stamp": null
        },
        "by_category": [
            {
                "category": "Property",
                "balance": 82042.54,
                "latest_balance_time_stamp": "2015-06-16T00:00:00.000Z",
                "subcategories": [
                    {
                        "subcategory": "other",
                        "balance": 82042.54,
                        "latest_balance_time_stamp": "2015-06-16T00:00:00.000Z"
                    }
                ]
            },
            {
                "category": "Loan",
                "balance": 56875.25,
                "latest_balance_time_stamp": "2015-06-16T00:00:00.000Z",
                "subcategories": [
                    {
                        "subcategory": "Mortgage",
                        "balance": 56875.25,
                        "latest_balance_time_stamp": "2015-06-16T00:00:00.000Z"
```

NAV ☰


```
                    }
                ]
            },
            {
                "category": "Banking",
                "balance": 90818.39,
                "latest_balance_time_stamp": "2015-06-16T00:00:00.000Z",
                "subcategories": [
                    {
                        "subcategory": "Savings",
                        "balance": 53246.2,
                        "latest_balance_time_stamp": "2015-06-16T00:00:00.000Z"
                    },
                    {
                        "subcategory": "Checking",
                        "balance": 37572.19,
                        "latest_balance_time_stamp": "2015-06-16T00:00:00.000Z"
                    }
                ]
            }
        ]
    },
    "change": {
        "analysis_start": "2015-06-15",
        "analysis_end": "2015-06-16",
        "total_assets": {
            "1_day": {
                "value": 0.0,
                "percentage": 0.0
            },
            "total": {
                "value": 0.0,
                "percentage": 0.0
            }
        },
        "total_liabilities": {
            "1_day": {
                "value": 0.0,
                "percentage": 0.0
            },
            "total": {
                "value": 0.0,
                "percentage": 0.0
            }
        },
        "net_worth": {
```

NAV ☰


```
                "1_day": {
                    "value": 0.0,
                    "percentage": 0.0
                },
                "total": {
                    "value": 0.0,
                    "percentage": 0.0
                }
            },
            "investable_cash": {
                "1_day": {
                    "value": 0.0,
                    "percentage": 0.0
                },
                "total": {
                    "value": 0.0,
                    "percentage": 0.0
                }
            },
            "retirement_savings": {
                "1_day": {
                    "value": 0.0,
                    "percentage": 0.0
                },
                "total": {
                    "value": 0.0,
                    "percentage": 0.0
                }
            }
        },
        "history": [
            {
                "date": "2015-06-15",
                "total_assets": 172860.93,
                "total_liabilities": 56875.25,
                "net_worth": 115985.68,
                "investable_cash": 90818.39,
                "retirement_savings": 0.0,
                "by_category": [
                    {
                        "category": "Property",
                        "balance": 82042.54,
                        "subcategories": [
                            {
                                "subcategory": "other",
                                "balance": 82042.54
```

NAV


```
                }
            ]
        },
        {
            "category": "Loan",
            "balance": 56875.25,
            "subcategories": [
                {
                    "subcategory": "Mortgage",
                    "balance": 56875.25
                }
            ]
        },
        {
            "category": "Banking",
            "balance": 90818.39,
            "subcategories": [
                {
                    "subcategory": "Savings",
                    "balance": 53246.2
                },
                {
                    "subcategory": "Checking",
                    "balance": 37572.19
                }
            ]
        }
    ]
},
{
    "date": "2015-06-16",
    "total_assets": 172860.93,
    "total_liabilities": 56875.25,
    "net_worth": 115985.68,
    "investable_cash": 90818.39,
    "retirement_savings": 0.0,
    "by_category": [
        {
            "category": "Property",
            "balance": 82042.54,
            "subcategories": [
                {
                    "subcategory": "other",
                    "balance": 82042.54
                }
            ]
```

NAV


```
                    },
                    {
                        "category": "Loan",
                        "balance": 56875.25,
                        "subcategories": [
                            {
                                "subcategory": "Mortgage",
                                "balance": 56875.25
                            }
                        ]
                    },
                    {
                        "category": "Banking",
                        "balance": 90818.39,
                        "subcategories": [
                            {
                                "subcategory": "Savings",
                                "balance": 53246.2
                            },
                            {
                                "subcategory": "Checking",
                                "balance": 37572.19
                            }
                        ]
                    }
                ]
            }
        ],
        "audit_log_id": "6ff9c751-3745-48ec-844f-683590d8a658"
    }
```


RESPONSE

Field	Description
`currency_code`	Currency code associated with monetary response values.
`snapshot`	A snapshot of the client's financial picture as of the most recent available date within the date range.
`total_assets`	Total assets of the client.

NAV

Field	Description
`balance`	Value of the total assets.
`latest_balance_time_stamp`	Date and time of the total assets record.
`total_liabilities`	Total liabilities of the client.
`balance`	Value of the total liabilities.
`latest_balance_time_stamp`	Date and time of the total liabilities record.
`net_worth`	Net worth of the client.
`balance`	Value of the net worth.
`latest_balance_time_stamp`	Date and time of the net worth record.
`retirement_savings`	Retirement savings of the client.
`balance`	Value of the retirement savings.
`latest_balance_time_stamp`	Date and time of the retirement savings record.
`investable_cash`	Investable cash of the client.
`balance`	Value of the investable cash.
`latest_balance_time_stamp`	Date and time of the investable cash record.

NAV

Field	Description
`by_category`	List of all aggregation accounts and balances by category and subcategory.
`category`	Category of the aggregation accounts. These accounts are dependent on the `category` field within the Nucleus Aggregation Account.
`balance`	Total balance for this category.
`latest_balance_time_stamp`	Date and time of the balance record for this category.
`subcategories`	List of subcategories within the category and their respective balances.
`subcategory`	Subcategory of the aggregation accounts. These accounts are dependent on the `subcategory` fields within the Nucleus Aggregation Account.
`balance`	Total balance for this subcategory.
`latest_balance_time_stamp`	Date and time of the balance record for this subcategory.
`change`	Change records of the client.
`analysis_start`	Start date of available data used in the change analysis.
`analysis_end`	End date of available data used in the change analysis.
`{total_assets, total_liabilities, net_worth}`	A map of the below change records will be provided for each of the following: `total_assets`, `total_liabilities`, `net_worth`, `retirement_savings`, and `investable_cash`.
`1-day`	1-day change record as of the analysis end

NAV

Field	Description
	date.
`value`	1-day value change as of the analysis end date.
`percentage`	1-day percentage change as of the analysis end date.
`7-day`	7-day change record as of the analysis end date.
`value`	7-day value change as of the analysis end date.
`percentage`	7-day percentage change as of the analysis end date.
`30-day`	30-day change record as of the analysis end date.
`value`	30-day value change as of the analysis end date.
`percentage`	30-day percentage change as of the analysis end date.
`60-day`	60-day change record as of the analysis end date.
`value`	60-day value change as of the analysis end date.
`percentage`	60-day percentage change as of the analysis end date.
`90-day`	90-day change record as of the analysis end date.
`value`	90-day value change as of the analysis end date.
`percentage`	90-day percentage change as of the analysis end date.

NAV

Field	Description
`180-day`	180-day change record as of the analysis end date.
`value`	180-day value change as of the analysis end date.
`percentage`	180-day percentage change as of the analysis end date.
`365-day`	365-day change record as of the analysis end date.
`value`	365-day value change as of the analysis end date.
`percentage`	365-day percentage change as of the analysis end date.
`total`	Change record over the entire analysis date range.
`value`	Total value change over the entire analysis date range.
`percentage`	Total percentage change over the entire analysis date range.
`history`	A historical record of the client's financials for each date within the date range. This is an array of maps, with each map corresponding to a date and containing the following fields:
`date`	The date of the financial picture record.
`total_assets`	Total assets of the client on this date.
`total_liabilities`	Total liabilities of the client on this date.
`net_worth`	Net worth of the client on this date.

NAV

Field	Description
`retirement_savings`	Retirement savings of the client on this date.
`investable_cash`	Investable cash of the client on this date.
`by_category`	List of all aggregation accounts and balances on this date by category and subcategory.
`category`	Category of the aggregation accounts. These accounts are dependent on the `category` field within the Nucleus Aggregation Account.
`balance`	Total balance for this category on this date.
`subcategories`	List of subcategories within the category and their respective balances on this date.
`subcategory`	Subcategory of the aggregation accounts. These accounts are dependent on the `subcategory` fields within the Nucleus Aggregation Account.
`balance`	Total balance for this subcategory on this date.
`audit_log_id`	The unique id of the Audit Log record created in Nucleus. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Household`
- `Client`
- `Aggregation Account`
- `Aggregation Account Balance`
- `Account Categories`

NAV ☰

Recurring Transaction Analysis

Recurring Transaction Analysis combs through a client's transaction history to automatically identify recurring patterns. These patterns often represent retail subscriptions and other kinds of repeated payments that a client makes, such as a gym membership or a content subscription. This analysis matches transactions against each other based on a common merchant, amount, and time interval. It also includes built-in controls to help capture transaction patterns that may deviate over time. The resulting output includes summary-level information across all recurring transactions as well an optional drill-down into each individual recurring transaction stream.

HTTP REQUEST

`POST /recurring_transaction_analysis`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "client_id": "579f5984-7281-4d60-9690-20590c7ccaa7",
        "scope": "all",
        "currency_code": "USD",
        "start_date": "2017-01-01",
        "end_date": "2020-08-15",
        "amount_deviation_threshold": 0.25,
        "interval_deviation_threshold": 5,
        "analyze_transactions": true,
        "show_recurring_details": true
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/recurring_transaction_
```

ARGUMENTS

Parameter	Description
`client_id` *UUID, conditional requirement*	The ID of a Nucleus Client used to source transaction data for the given `scope`, when one of `account_ids` or `aggregation_account_ids` is not provided. Conditionally required if one of `aggregation_account_ids` or `account_ids` is not passed in the request.

NAV

Parameter	Description
`aggregation_account_ids` *array[UUID], conditional requirement*	An array of ID(s) of Nucleus Aggregation Accounts to be used when `scope` is `all` or `external`. Conditionally required if one of `client_id` or `account_ids` is not passed in the request.
`account_ids` *array[UUID], conditional requirement*	An array of ID(s) of Nucleus Accounts to be used when `scope` is `all` or `internal`. Conditionally required if one of `client_id` or `aggregation_account_ids` is not passed in the request.
`scope` *string*	The scope of data to be considered. Value must be one of the following: `all`, `external`, `internal`. Defaults to `all`, which will consider both `internal` transactions, or Portfolio Transactions, and `external` transactions, or Aggregation Account Transactions.
`transaction_status_scope` *array[string]*	If populated, only transactions whose `status` matches one of the array values are considered in the analysis. Defaults to `null`, meaning all transactions are considered regardless of the `status`.
`currency_code` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Only records with this currency code will be considered. Defaults to USD if `currency_conversion` is not provided. If `currency_conversion` is provided, all origin currencies will be considered by default. See Currencies.
`currency_conversion` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Records will be converted from their original currency to the one indicated here. Conversions are supported both to and from the following currency codes: USD, GBP, EUR, AUD, CAD, CHF. See Currencies.
`start_date` *date*	Start date of the analysis. Defaults to earliest date there is data.
`end_date` *date*	End date of the analysis. Defaults to latest date there is data.
`amount_deviation_threshold` *float*	Allowable deviation threshold for transaction amount, measured on a relative percentage basis. If this value is > 0, transaction amounts need not be exactly the same in order to be considered recurring. Must be between 0.0 and 1.0, inclusive. Defaults to 0.05.
`interval_deviation_threshold` *integer*	Allowable deviation threshold for transaction interval, measured in days. If this value is > 0, time intervals between transactions need

NAV

Parameter	Description
	not be exactly the same in order to be considered recurring. Must be greater than or equal to 0. Defaults to 3.
`analyze_transactions` *boolean*	If `true`, dynamically analyze transactions. If `false`, only consider transactions that have the `is_recurring` attribute. Defaults to `true`.
`flag_transactions` *boolean*	If `true`, one or more Bulk Update jobs will be executed to update the `is_recurring` parameter from `false` to `true` for any transactions identified as recurring in this analysis. Only applicable when `analyze_transactions` is set to `true`. Defaults to `false`, meaning no transactions will be updated.
`show_recurring_details` *boolean*	If `true`, return advanced details about each recurring transaction stream in the analysis. Defaults to `false`.
`only_cleansed` *boolean*	If `true`, only Portfolio Transactions with the `is_cleansed` parameter set to `true` will be considered. Defaults to `false`.

Example Response


```
{
    "analysis_start": "2017-01-01",
    "analysis_end": "2020-08-15",
    "currency_code": "USD",
    "total_amount_spent": 969.58,
    "by_merchant": [
        {
            "merchant": "Citibike",
            "amount_spent": 400.0
        },
        {
            "merchant": "Netflix",
            "amount_spent": 41.97
        },
        {
            "merchant": "Crunch",
            "amount_spent": 313.96
        },
        {
```

NAV


```
                "merchant": "Freshly",
                "amount_spent": 213.65
            }
        ],
        "by_category": [
            {
                "category": "Travel",
                "amount_spent": 400.0,
                "subcategories": [
                    {
                        "subcategory": "Public Transportation",
                        "amount_spent": 400.0
                    }
                ]
            },
            {
                "category": "Entertainment",
                "amount_spent": 41.97,
                "subcategories": [
                    {
                        "subcategory": "Streaming",
                        "amount_spent": 41.97
                    }
                ]
            },
            {
                "category": "Health & Fitness",
                "amount_spent": 313.96,
                "subcategories": [
                    {
                        "subcategory": "Fitness Centers & Gyms",
                        "amount_spent": 313.96
                    }
                ]
            },
            {
                "category": "Food & Dining",
                "amount_spent": 213.65,
                "subcategories": [
                    {
                        "subcategory": "Meal Plans",
                        "amount_spent": 213.65
                    }
                ]
            }
        ],
```

NAV


```
        "recurring_details": [
            {
                "merchant": "Citibike",
                "category": "Travel",
                "subcategory": "Public Transportation",
                "amount": 125.0,
                "interval": 366,
                "number_of_transactions": 3,
                "earliest_transaction_date": "2017-09-26T00:00:00.000+0000",
                "latest_transaction_date": "2019-09-29T00:00:00.000+0000",
                "projected_transaction_date": "2020-09-29T00:00:00.000+0000"
            },
            {
                "merchant": "Netflix",
                "category": "Entertainment",
                "subcategory": "Streaming",
                "amount": 13.99,
                "interval": 30,
                "number_of_transactions": 3,
                "earliest_transaction_date": "2020-04-15T00:00:00.000+0000",
                "latest_transaction_date": "2020-06-15T00:00:00.000+0000",
                "projected_transaction_date": "2020-07-15T00:00:00.000+0000"
            },
            {
                "merchant": "Crunch",
                "category": "Health & Fitness",
                "subcategory": "Fitness Centers & Gyms",
                "amount": 79.25,
                "interval": 31,
                "number_of_transactions": 4,
                "earliest_transaction_date": "2019-07-08T00:00:00.000+0000",
                "latest_transaction_date": "2019-10-09T00:00:00.000+0000",
                "projected_transaction_date": "2019-11-09T00:00:00.000+0000"
            },
            {
                "merchant": "Freshly",
                "category": "Food & Dining",
                "subcategory": "Meal Plans",
                "amount": 73.12,
                "interval": 7,
                "number_of_transactions": 3,
                "earliest_transaction_date": "2020-07-01T00:00:00.000+0000",
                "latest_transaction_date": "2020-07-15T00:00:00.000+0000",
                "projected_transaction_date": "2020-07-22T00:00:00.000+0000"
            }
```

NAV

```
        ]
    }
```

RESPONSE

Field	Description
`analysis_start`	Start date of the analysis.
`analysis_end`	End date of the analysis.
`currency_code`	Currency code associated with monetary response values.
`total_amount_spent`	Total expenditure on recurring transactions during the analysis period.
`by_merchant`	Array of recurring expenditure by merchant.
`merchant`	Name of the merchant.
`amount_spent`	Total expenditure for the merchant during the analysis period.
`by_category`	Array of recurring expenditure by category.
`category`	Name of the category.
`amount_spent`	Total expenditure for the category during the analysis period.
`subcategories`	Array of recurring expenditure by subcategory.
`subcategory`	Name of the subcategory.
`amount_spent`	Total expenditure for the subcategory during the analysis period.
`recurring_details`	Array of recurring transaction details, where each entry represents a stream of transactions.

NAV

Field	Description
`merchant`	Name of the merchant.
`category`	Category of the recurring transaction.
`subcategory`	Subcategory of the recurring transaction.
`amount`	Median amount of the recurring transaction.
`interval`	Median number of days between each transaction.
`number_of_transactions`	Number of transactions in the stream.
`earliest_transaction_date`	Earliest recurring transaction date in the stream.
`latest_transaction_date`	Latest recurring transaction date in the stream.
`projected_transaction_date`	Projected next recurring transaction date in the stream.
`bulk_update_ids`	An array of one or more `bulk_ids` returned from the Bulk Update jobs. Only returned if `flag_transactions` is set to `true` .

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Account`
- `Portfolio`
- `Portfolio Transaction`
- `Aggregation Account`
- `Aggregation Account Transaction`

NAV

Spending Analysis

A detailed and configurable analysis of spending

HTTP REQUEST

```
POST /spending_analysis
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "business_ids": [
            "b589ad60-86ad-42a0-b4fc-4e9b1587ff8e"
        ],
        "currency_code": "USD",
        "scope": "all",
        "card_status_scope": [
            "activated"
        ],
        "transaction_status_scope": [
            "completed",
            "pending"
        ],
        "transaction_category_scope": [
            "e594f949-6f92-11eb-922d-124ab2ff7f93"
        ],
        "merchant_scope": [
            "0a7b62b4-54f3-4cef-afd9-c695d1d28740"
        ],
        "frequency_unit": "monthly",
        "frequency": 1,
        "start_date": "2021-02-01",
        "end_date": "2021-04-26",
        "show_by_period": true,
        "show_by_category": true,
        "show_by_merchant": true,
        "only_cleansed": false,
        "only_active_clients": false
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/spending_analysis"
```

NAV

ARGUMENTS

Parameter	Description
`business_ids` *array[UUID], conditional requirement*	ID(s) of the Nucleus Business(es) used to source transaction data. All transactions linked to each business will be considered. Conditionally required if one of `household_ids`, `client_ids`, `card_ids`, `aggregation_account_ids`, or `account_ids` is not provided.
`household_ids` *array[UUID], conditional requirement*	ID(s) of the Nucleus Household(s) used to source transaction data. All transactions linked to each household will be considered. Conditionally required if one of `business_ids`, `client_ids`, `card_ids`, `aggregation_account_ids`, or `account_ids` is not provided.
`client_ids` *array[UUID], conditional requirement*	ID(s) of the Nucleus Client(s) used to source transaction data. All transactions linked to each client will be considered. Conditionally required if one of `business_ids`, `household_ids`, `card_ids`, `aggregation_account_ids`, or `account_ids` is not provided.
`card_ids` *array[UUID], conditional requirement*	ID(s) of the Nucleus Card(s) used to source transaction data. All transactions linked to each card will be considered. Conditionally required if one of `business_ids`, `household_ids`, `client_ids`, `aggregation_account_ids`, or `account_ids` is not provided.
`aggregation_account_ids` *array[UUID], conditional requirement*	ID(s) of the Nucleus Aggregation Account(s) used to source transaction data. All transactions linked to each account will be considered. Conditionally required if one of `business_ids`, `household_ids`, `client_ids`, `card_ids`, or `account_ids` is not provided.
`account_ids` *array[UUID], conditional requirement*	ID(s) of the Nucleus Account(s) used to source transaction data. All transactions linked to each account will be considered. Conditionally required if one of `business_ids`, `household_ids`, `client_ids`, `card_ids`, or `aggregation_account_ids` is not provided.
`currency_code` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Only records with this currency code will be considered. Defaults to USD if `currency_conversion` is not provided. If `currency_conversion` is provided, all origin currencies will be considered by default. See Currencies.
`currency_conversion` *string*	The alphabetic currency code used to conduct the analysis, limited to 3 characters. Records will be converted from their original currency to the one indicated here. Conversions are supported both to and from

NAV

Parameter	Description
	the following currency codes: USD, GBP, EUR, AUD, CAD, CHF. See Currencies.
`scope` *string*	The scope of data to be considered. Value must be one of the following: `all`, `external`, `internal`, or `cards`. Defaults to `all`, which will consider both `internal` transactions, or Portfolio Transactions, and `external` transactions, or Aggregation Account Transactions. Passing a value of `cards` will return a subset of `internal` data, which is only Portfolio Transactions that are linked to cards. Only applicable when neither `card_ids`, `account_ids`, or `aggregation_account_ids` is provided.
`card_status_scope` *array[string]*	If populated, only cards whose `status` matches one of the array values are considered in the analysis. Defaults to `null`, meaning transaction data from all cards are considered regardless of the `status`. Only applicable when `scope` is set to `cards`.
`transaction_status_scope` *array[string]*	If populated, only transactions whose `status` matches one of the array values are considered in the analysis. Defaults to `null`, meaning all transactions are considered regardless of the `status`.
`transaction_category_scope` *array[UUID]*	If populated, only transactions whose `transaction_category_id` matches one of the array values are considered in the analysis. A `transaction_category_id` is considered primary if it has a `subcategory` of `null`, meaning it represents a high-level category grouping. If a primary `transaction_category_id` is passed, then this scope will also include any `transaction_category_ids` with the same `category` value and different `subcategory` values. Defaults to `null`, meaning all transactions are considered regardless of the `transaction_category_id`. See Transaction Categories.
`merchant_scope` *array[UUID]*	If populated, only transactions whose `merchant_id` matches one of the array values are considered in the analysis. Defaults to `null`, meaning all transactions are considered regardless of the `merchant_id`. See Merchants.
`frequency_unit` *string*	Frequency unit of the analysis. Value may be `daily`, `weekly`, `bi-weekly`, `monthly`, `quarterly`, or `annually`. Defaults to `monthly`.
`frequency` *integer*	Number of `frequency_unit` between each period. For example, if the `frequency_unit` is `weekly` and the `frequency` is `2`, this means each period spans two weeks. Default is `1`.

NAV

Parameter	Description
`as_of_date` *date*	Reference date of the analysis that determines the starting point of the `lookback_periods`. Defaults to today's date. Not applicable if a `start_date` and `end_date` are provided.
`lookback_periods` *integer*	Number of lookback periods to analyze. Each period length is defined by the combination of `frequency` and `frequency_unit` provided, beginning at the `as_of_date` provided. For example, if `frequency_unit` is `monthly`, `frequency` is `1`, and `lookback_periods` is `3`, then the calendar month through the `as_of_date`, plus the previous 3 full calendar months, will be analyzed. Defaults to `0`, meaning that only the period containing the `as_of_date` is analyzed. Not applicable if a `start_date` and `end_date` are provided.
`start_date` *date*	Start date of the analysis. Conditionally required if `end_date` is passed. Passing `start_date` and `end_date` will override `as_of_date` and `lookback_periods`.
`end_date` *date*	End date of the analysis. Conditionally required if `start_date` is passed. Passing `start_date` and `end_date` will override `as_of_date` and `lookback_periods`.
`show_by_period` *boolean*	If `true`, a detailed history of spending will be returned over each of the periods determined by `frequency_unit` and `frequency`, over the entire analysis period determined by either `as_of_date` and `lookback_periods` or `start_date` and `end_date`. Defaults to `false`.
`show_by_category` *boolean*	If `true`, return breakdowns of spending by category and subcategory. Defaults to `false`.
`show_by_merchant` *boolean*	If `true`, return breakdowns of spending by merchant. Defaults to `false`.
`only_cleansed` *boolean*	If `true`, only Portfolio Transactions with the `is_cleansed` parameter set to `true` will be considered. Defaults to `false` so that all transactions are considered.
`only_active_clients` *boolean*	If `true`, only Clients with the `is_active` parameter set to `true` will be considered. Defaults to `false` so that all clients are considered.

NAV III

Example Response


```
{
    "analysis_start": "2021-02-01",
    "analysis_end": "2021-04-26",
    "currency_code": "USD",
    "summary": {
        "number_of_transactions": 13,
        "amount_spent": 2267.7,
        "mean_transaction_amount": 174.44,
        "by_category": [
            {
                "category": "Automotive",
                "number_of_transactions": 13,
                "amount_spent": 2267.7,
                "mean_transaction_amount": 174.44,
                "weight": 1.0,
                "by_merchant": [
                    {
                        "merchant": "Pep Boys",
                        "merchant_id": "0a7b62b4-54f3-4cef-afd9-c695d1d28740",
                        "number_of_transactions": 13,
                        "amount_spent": 2267.7,
                        "mean_transaction_amount": 174.44,
                        "weight": 1.0
                    }
                ],
                "subcategories": [
                    {
                        "subcategory": "Other",
                        "transaction_category_id": "e594f949-6f92-11eb-922d-124ab2ff7f
                        "number_of_transactions": 13,
                        "amount_spent": 2267.7,
                        "mean_transaction_amount": 174.44,
                        "weight": 1.0,
                        "by_merchant": [
                            {
                                "merchant": "Pep Boys",
                                "merchant_id": "0a7b62b4-54f3-4cef-afd9-c695d1d28740",
                                "number_of_transactions": 13,
                                "amount_spent": 2267.7,
                                "mean_transaction_amount": 174.44,
                                "weight": 1.0
                            }
```

NAV III


```
                        ]
                    }
                ]
            }
        ],
        "by_merchant": [
            {
                "merchant": "Pep Boys",
                "merchant_id": "0a7b62b4-54f3-4cef-afd9-c695d1d28740",
                "number_of_transactions": 13,
                "amount_spent": 2267.7,
                "mean_transaction_amount": 174.44,
                "weight": 1.0,
                "by_category": [
                    {
                        "category": "Automotive",
                        "number_of_transactions": 13,
                        "amount_spent": 2267.7,
                        "mean_transaction_amount": 174.44,
                        "weight": 1.0,
                        "subcategories": [
                            {
                                "subcategory": "Other",
                                "transaction_category_id": "e594f949-6f92-11eb-922d-12
                                "number_of_transactions": 13,
                                "amount_spent": 2267.7,
                                "mean_transaction_amount": 174.44,
                                "weight": 1.0
                            }
                        ]
                    }
                ]
            }
        ]
    },
    "by_period": [
        {
            "period_start": "2021-02-01",
            "period_end": "2021-02-28",
            "analysis_start": "2021-02-01",
            "analysis_end": "2021-02-28",
            "number_of_transactions": 1,
            "amount_spent": 22.55,
            "mean_transaction_amount": 22.55,
            "by_category": [
                {
```

NAV


```
                "category": "Automotive",
                "number_of_transactions": 1,
                "amount_spent": 22.55,
                "mean_transaction_amount": 22.55,
                "weight": 1.0,
                "by_merchant": [
                    {
                        "merchant": "Pep Boys",
                        "merchant_id": "0a7b62b4-54f3-4cef-afd9-c695d1d28740",
                        "number_of_transactions": 1,
                        "amount_spent": 22.55,
                        "mean_transaction_amount": 22.55,
                        "weight": 1.0
                    }
                ],
                "subcategories": [
                    {
                        "subcategory": "Other",
                        "transaction_category_id": "e594f949-6f92-11eb-922d-124ab2
                        "number_of_transactions": 1,
                        "amount_spent": 22.55,
                        "mean_transaction_amount": 22.55,
                        "weight": 1.0,
                        "by_merchant": [
                            {
                                "merchant": "Pep Boys",
                                "merchant_id": "0a7b62b4-54f3-4cef-afd9-c695d1d287
                                "number_of_transactions": 1,
                                "amount_spent": 22.55,
                                "mean_transaction_amount": 22.55,
                                "weight": 1.0
                            }
                        ]
                    }
                ]
            }
        ],
        "by_merchant": [
            {
                "merchant": "Pep Boys",
                "merchant_id": "0a7b62b4-54f3-4cef-afd9-c695d1d28740",
                "number_of_transactions": 1,
                "amount_spent": 22.55,
                "mean_transaction_amount": 22.55,
                "weight": 1.0,
                "by_category": [
```

NAV III


```
                    {
                        "category": "Automotive",
                        "number_of_transactions": 1,
                        "amount_spent": 22.55,
                        "mean_transaction_amount": 22.55,
                        "weight": 1.0,
                        "subcategories": [
                            {
                                "subcategory": "Other",
                                "transaction_category_id": "e594f949-6f92-11eb-922
                                "number_of_transactions": 1,
                                "amount_spent": 22.55,
                                "mean_transaction_amount": 22.55,
                                "weight": 1.0
                            }
                        ]
                    }
                ]
            }
        ]
    },
    {
        "period_start": "2021-03-01",
        "period_end": "2021-03-31",
        "analysis_start": "2021-03-01",
        "analysis_end": "2021-03-31",
        "number_of_transactions": 12,
        "amount_spent": 2245.15,
        "mean_transaction_amount": 187.1,
        "by_category": [
            {
                "category": "Automotive",
                "number_of_transactions": 12,
                "amount_spent": 2245.15,
                "mean_transaction_amount": 187.1,
                "weight": 1.0,
                "by_merchant": [
                    {
                        "merchant": "Pep Boys",
                        "merchant_id": "0a7b62b4-54f3-4cef-afd9-c695d1d28740",
                        "number_of_transactions": 12,
                        "amount_spent": 2245.15,
                        "mean_transaction_amount": 187.1,
                        "weight": 1.0
                    }
                ],
```

NAV


```
                "subcategories": [
                    {
                        "subcategory": "Other",
                        "transaction_category_id": "e594f949-6f92-11eb-922d-124ab2
                        "number_of_transactions": 12,
                        "amount_spent": 2245.15,
                        "mean_transaction_amount": 187.1,
                        "weight": 1.0,
                        "by_merchant": [
                            {
                                "merchant": "Pep Boys",
                                "merchant_id": "0a7b62b4-54f3-4cef-afd9-c695d1d287
                                "number_of_transactions": 12,
                                "amount_spent": 2245.15,
                                "mean_transaction_amount": 187.1,
                                "weight": 1.0
                            }
                        ]
                    }
                ]
            }
        ],
        "by_merchant": [
            {
                "merchant": "Pep Boys",
                "merchant_id": "0a7b62b4-54f3-4cef-afd9-c695d1d28740",
                "number_of_transactions": 12,
                "amount_spent": 2245.15,
                "mean_transaction_amount": 187.1,
                "weight": 1.0,
                "by_category": [
                    {
                        "category": "Automotive",
                        "number_of_transactions": 12,
                        "amount_spent": 2245.15,
                        "mean_transaction_amount": 187.1,
                        "weight": 1.0,
                        "subcategories": [
                            {
                                "subcategory": "Other",
                                "transaction_category_id": "e594f949-6f92-11eb-922
                                "number_of_transactions": 12,
                                "amount_spent": 2245.15,
                                "mean_transaction_amount": 187.1,
                                "weight": 1.0
                            }
```

NAV


```
                                ]
                            }
                        ]
                    }
                ]
            },
            {
                "period_start": "2021-04-01",
                "period_end": "2021-04-30",
                "analysis_start": "2021-04-01",
                "analysis_end": "2021-04-26",
                "number_of_transactions": 0,
                "amount_spent": 0,
                "mean_transaction_amount": 0,
                "by_category": [],
                "by_merchant": []
            }
        ]
    }
```


RESPONSE

Field	Description
`analysis_start`	Start date of the analysis.
`analysis_end`	End date of the analysis.
`currency_code`	Currency code associated with monetary response values.
`summary`	Summary of spending over the analysis period.
`number_of_transactions`	Number of transactions over the analysis period.
`amount_spent`	Amount spent over the analysis period.
`mean_transaction_amount`	Average transaction amount over the analysis period.
`by_category`	List of spending information separated by category and their relative subcategories. Only returned when

NAV

Field	Description
	`show_by_category` is set to `true`.
`category`	Category as defined in the Nucleus transactions.
`number_of_transactions`	Number of transactions over the analysis period for the given category.
`amount_spent`	Amount spent over the analysis period for the given category.
`mean_transaction_amount`	Average transaction amount over the analysis period for the given category.
`weight`	Proportion of spending over the analysis period related to this category.
`by_merchant`	List of spending information for the category, separated by merchant. Only returned when `show_by_merchant` is set to `true`.
`merchant`	Merchant as defined in the Nucleus transactions.
`merchant_id`	ID of the merchant record for the merchant. See Merchants.
`number_of_transactions`	Number of transactions over the analysis period for the given category and merchant.
`amount_spent`	Amount spent over the analysis period for the given category and merchant.
`mean_transaction_amount`	Average transaction amount over the analysis period for the given category and merchant.
`weight`	Proportion of spending over the analysis period related to this category and merchant.
`subcategories`	List of subcategories for the given category.

NAV

Field	Description
`subcategory`	Subcategory as defined in the Nucleus transactions.
`transaction_category_id`	ID of the transaction category record for the given category/subcategory combination. See Transaction Categories.
`number_of_transactions`	Number of transactions over the analysis period for the given subcategory.
`amount_spent`	Amount spent over the analysis period for the given subcategory.
`mean_transaction_amount`	Average transaction amount over the analysis period for the given subcategory.
`weight`	Proportion of spending over the analysis period related to this subcategory.
`by_merchant`	List of spending information for the subcategory, separated by merchant. Only returned when `show_by_merchant` is set to `true`.
`merchant`	Merchant as defined in the Nucleus transactions.
`merchant_id`	ID of the merchant record for the merchant. See Merchants.
`number_of_transactions`	Number of transactions over the analysis period for the given subcategory and merchant.
`amount_spent`	Amount spent over the analysis period for the given subcategory and merchant.
`mean_transaction_amount`	Average transaction amount over the analysis period for the given subcategory and merchant.
`weight`	Proportion of spending over the analysis period related to this subcategory and merchant.
`by_merchant`	List of spending information separated by merchant. Only

NAV

Field	Description
	returned when `show_by_merchant` is set to `true`.
`merchant`	Merchant as defined in the Nucleus transactions.
`merchant_id`	ID of the merchant record for the merchant. See Merchants.
`number_of_transactions`	Number of transactions over the analysis period for the given merchant.
`amount_spent`	Amount spent over the analysis period for the given merchant.
`mean_transaction_amount`	Average transaction amount over the analysis period for the given merchant.
`weight`	Proportion of spending over the analysis period related to this merchant.
`by_category`	List of spending information for the merchant, separated by category and their relative subcategories. Only returned when `show_by_category` is set to `true`.
`category`	Category as defined in the Nucleus transactions.
`number_of_transactions`	Number of all transactions over the analysis period for the given merchant and category.
`amount_spent`	Amount spent over the analysis period for the given merchant and category.
`mean_transaction_amount`	Average transaction amount over the analysis period for the given merchant and category.
`weight`	Proportion of spending over the analysis period related to this merchant and category.
`subcategories`	List of subcategories for the given category.

NAV

Field	Description
`subcategory`	Subcategory as defined in the Nucleus transactions.
`transaction_category_id`	ID of the transaction category record for the given category/subcategory combination. See Transaction Categories
`number_of_transactions`	Number of transactions over the analysis period for the given merchant and subcategory.
`amount_spent`	Amount spent over the analysis period for the given merchant and subcategory.
`mean_transaction_amount`	Average transaction amount over the analysis period for the given merchant and subcategory.
`weight`	Proportion of spending over the analysis period related to this merchant and subcategory.
`by_period`	Historical spending data, with frequency dependent on the values of `frequency_unit` and `frequency` passed in the request. Each entry has the fields shown below.
`period_start`	Start date of the period.
`period_end`	End date of the period.
`analysis_start`	Start date of the analysis period.
`analysis_end`	End date of the analysis period.
`number_of_transactions`	Number of transactions over the period.
`amount_spent`	Amount spent over the period.
`mean_transaction_amount`	Average transaction amount over the period.

NAV

Field	Description
`by_category`	List of spending information for the merchant, separated by category and their relative subcategories. Only returned when `show_by_category` is set to `true`.
`category`	Category as defined in the Nucleus transactions.
`number_of_transactions`	Number of transactions over the period for the given category.
`amount_spent`	Amount spent over the period for the given category.
`mean_transaction_amount`	Average transaction amount over the period for the given category.
`weight`	Proportion of spending over the period related to this category.
`by_merchant`	List of spending information for the category, separated by merchant. Only returned when `show_by_merchant` is set to `true`.
`merchant`	Merchant as defined in the Nucleus transactions.
`merchant_id`	ID of the merchant record for the merchant. See Merchants
`number_of_transactions`	Number of transactions over the period for the given category and merchant.
`amount_spent`	Amount spent over the period for the given category and merchant.
`mean_transaction_amount`	Average transaction amount over the period for the given category and merchant.
`weight`	Proportion of spending over the period related to this category and merchant.

NAV

Field	Description
`subcategories`	List of subcategories for the given category.
`subcategory`	Subcategory as defined in the Nucleus transactions.
`transaction_category_id`	ID of the transaction category record for the given category/subcategory combination. See Transaction Categories.
`number_of_transactions`	Number of transactions over the period for the given subcategory.
`amount_spent`	Amount spent over the period for the given subcategory.
`mean_transaction_amount`	Average transaction amount over the period for the given category and merchant.
`weight`	Proportion of spending over the period related to this category and merchant.
`by_merchant`	List of spending information for the subcategory, separated by merchant. Only returned when `show_by_merchant` is set to `true`.
`merchant`	Merchant as defined in the Nucleus transactions.
`merchant_id`	ID of the merchant record for the merchant. See Merchants.
`number_of_transactions`	Number of transactions over the period for the given subcategory and merchant.
`amount_spent`	Amount spent over the period for the given subcategory and merchant.
`mean_transaction_amount`	Average transaction amount over the period for the given subcategory and merchant.
`weight`	Proportion of spending over the period related to this

NAV

Field	Description
	subcategory and merchant.
`by_merchant`	List of spending information separated by merchant. Only returned when `show_by_merchant` is set to `true`.
`merchant`	Merchant as defined in the Nucleus transactions.
`merchant_id`	ID of the merchant record for the merchant. See Merchants.
`number_of_transactions`	Number of transactions over the period for the given merchant.
`amount_spent`	Amount spent over the period for the given merchant.
`mean_transaction_amount`	Average transaction amount over the period for the given merchant.
`weight`	Proportion of spending over the period related to this merchant.
`by_category`	List of spending information for the merchant, separated by category and their relative subcategories. Only returned when `show_by_category` is set to `true`.
`category`	Category as defined in the Nucleus transactions.
`number_of_transactions`	Number of transactions over the period for the given merchant and category.
`amount_spent`	Amount spent over the period for the given merchant and category.
`mean_transaction_amount`	Average transaction amount over the period for the given merchant and category.
`weight`	Proportion of spending over the period related to this merchant and category.

NAV

Field	Description
`subcategories`	List of subcategories for the given merchant and category.
`subcategory`	Subcategory as defined in the Nucleus transactions.
`transaction_category_id`	ID of the transaction category record for the given category/subcategory combination. See Transaction Categories.
`number_of_transactions`	Number of transactions over the period for the given merchant and subcategory.
`amount_spent`	Amount spent over the period for the given merchant and subcategory.
`mean_transaction_amount`	Average transaction amount over the period for the given merchant and subcategory.
`weight`	Proportion of spending over the period related to this merchant and subcategory.

NUCLEUS DATA DEPENDENCIES

- `Business`
- `Household`
- `Client`
- `Card`
- `Account`
- `Portfolio`
- `Portfolio Transaction`
- `Aggregation Account`
- `Aggregation Account Transaction`
- `Merchant`
- `Transaction Category`

Portfolio Construction

NAV ☰

Diversification Score

This service is a scoring engine to assess the level of diversification exhibited by an investor's portfolio. The score compares the portfolio volatility to the volatility values of the underlying portfolio holdings, in order to derive a score from 0 to 100, where 0 indicates no diversification and 100 indicates the maximum theoretical diversification level.

HTTP REQUEST

```
POST /diversification_score
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "portfolio_tickers": [
            "XOM",
            "CMCSA",
            "AMZN",
            "GOOGL",
            "AGG"
        ],
        "portfolio_weights": [
            0.6,
            0.1,
            0.1,
            0.1,
            0.1
        ],
        "use_proxy_data": false
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/diversification_score"
```

ARGUMENTS

Parameter	Description
`portfolio_tickers` *array, conditional requirement*	Portfolio tickers, referencing securities defined in the Nucleus API.

NAV

Parameter	Description
`portfolio_weights` *array, conditional requirement*	Portfolio weights, corresponding to `portfolio_tickers`. Must sum to `1.0`.
`start_date` *date*	Start date used for ticker price history. Defaults to the earliest common date among `portfolio_tickers` prices.
`end_date` *date*	End date used for ticker price history. Defaults to the latest common date among `portfolio_tickers` prices.
`use_proxy_data` *boolean*	If `true`, incorporate proxy price data as defined at the Security level in the Nucleus API. Proxy data is merged with base security data to form a continuous price history. Defaults to `false`.
`market_data_source` *string*	Source of security price data to be used in the analysis. Value may be `nucleus` or `integration`. When value is `nucleus`, security price data is sourced from Nucleus Security Price. When value is `integration`, security price data is sourced from a configured Integration Market Data vendor. Defaults to `nucleus`.
`account_id` *UUID, conditional requirement*	The ID of a Nucleus account used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`aggregation_account_id` *UUID, conditional requirement*	The ID of a Nucleus aggregation account used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`allocation_id` *UUID, conditional requirement*	The ID of a Nucleus allocation used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`model_id` *UUID, conditional requirement*	The ID of a Nucleus model used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`portfolio_id` *UUID, conditional requirement*	The ID of a Nucleus portfolio used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is

NAV

Parameter	Description
	not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

Example Response

```
{
    "portfolio_standard_deviation": 0.15359097006164063,
    "holdings_standard_deviation": {
        "AGG": 0.03539292330703836,
        "AMZN": 0.30441576222578853,
        "CMCSA": 0.1830429265788811,
        "GOOGL": 0.2389454994916491,
        "XOM": 0.20299112408344339
    },
    "diversification_score": 23
}
```

RESPONSE

Field	Description
`portfolio_standard_deviation`	The annualized standard deviation of the overall portfolio.
`holdings_standard_deviation`	The annualized standard deviations of the individual portfolio holdings.
`diversification_score`	A score from 0 to 100, in which 0 indicates no diversification and 100 indicates the maximum theoretical diversification level.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

NAV
- Security
- Security Price
- Account
- Portfolio
- Portfolio Holding
- Model
- Model Holding
- Allocation
- Allocation Composition
- Aggregation Account
- Aggregation Account Holding

Mean-Variance Optimization

This service generates portfolios based on the principles of convex optimization. Securities are systematically weighted to maximize expected portfolio return for a given level of portfolio risk, subject to the defined constraints. Two outcomes can be achieved with this framework: target portfolio construction and efficient frontier construction. A target portfolio is a single optimal portfolio that adheres to a desired risk or return level, while an efficient frontier is a collection of optimal portfolios across a maximal range of risk and return levels.

HTTP REQUEST

`POST /mvo`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "tickers": ["VTI", "AGG", "EFA", "VWO", "GLD", "SHY"],
        "min_assets": 2,
        "w_config": {
          "w_min_major": 0.05,
          "w_max_major": 0.7,
          "w_min_minor": 0.05,
          "w_max_minor": 0.3,
          "cash_amount": 0.03
        },
        "w_asset_config": {
```

NAV ☰

```
    "US_Equities": 1.0,
    "Fixed_Income": 1.0,
    "Intl_Equities": 1.0,
    "EM_Equities": 1.0,
    "Commodities": 1.0
  },
  "sec_types": ["major", "major", "minor", "minor", "minor", "Cash"],
  "tgt_type": "risk",
  "tgt_val": 0.09,
  "start_date": "2015-01-01",
  "end_date": "2017-01-01",
  "use_proxy_data": false
}' "https://[sandbox][api].hydrogenplatform.com/proton/v1/mvo"
```

ARGUMENTS

Parameter	Description
`tickers` *array, required*	A list of tickers to consider during the optimization process, referencing securities defined in the Nucleus API. We recommend using no more than 25 tickers in a single optimization. Pre-screening a securities universe prior to calling the service will result in faster execution times as well as more meaningful results.
`min_assets` *integer, required*	The minimum number of portfolio assets, excluding the cash security (if applicable).
`w_config` *map, required*	Weight constraints for security types. Values that can be set are `w_min_major`, `w_max_major`, `w_min_minor`, `w_max_minor`, and `cash_amount`, which stipulate lower and upper bounds for minor and major securities, as well as a constant weight for cash securities. Minimums are joined with zero to form a binary constraint. This means that the minimum weight of securities of each type may either be 0 (i.e. not included in the final portfolio) or the stipulated minimum. For example, a `w_min_major` constraint of 5% indicates that if a `major` security is chosen during the optimization, its weight must be greater than or equal to 5%. If, however, the security is not chosen, its weight is also allowable at 0%.
`w_asset_config` *map*	Weight constraints for asset classes. Stipulates a maximum weight for each asset class represented in tickers. If an asset class is not found, the max constraint defaults to 100%.

NAV

Parameter	Description
`sec_types` *array, required*	A type for each security in `tickers`. Values may be `major`, `minor`, and `cash`. `major` securities are intended to be securities with more lenient weight thresholds. `minor` securities are intended to be securities with tighter weight thresholds. `cash` securities are constrained to a constant weight. `major` and `minor` designations interact with `w_config` to set distinct weight constraints for different kinds of securities. A maximum of one security may be labelled with the `cash` type.
`tgt_type` *string*	The target type for a target portfolio. If specified, a target portfolio is returned. Value may be `risk` or `return`. Risk targets place an upper bound on the allowed portfolio risk, while return targets place a lower bound on the allowed portfolio return. If excluded, a frontier of portfolios is returned.
`tgt_val` *float*	The target value for a target portfolio. Corresponds to `tgt_type`, and should indicate either an annualized portfolio return or an annualized portfolio standard deviation. If excluded, defaults to `0`.
`start_date` *date*	Start date for historical prices, represented in yyyy-mm-dd format. If excluded, defaults to today.
`end_date` *date*	End date for historical prices, represented in yyyy-mm-dd format. If excluded, defaults to today.
`use_proxy_data` *boolean*	If `true`, incorporate proxy price data as defined at the Security level in the Nucleus API. Proxy data is merged with base security data to form a continuous price history. Defaults to `false`.
`market_data_source` *string*	Source of security price data to be used in the analysis. Value may be `nucleus` or `integration`. When value is `nucleus`, security price data is sourced from Nucleus Security Price. When value is `integration`, security price data is sourced from a configured Integration Market Data vendor. Defaults to `nucleus`.

Example Response

```
{
    "target_portfolio": {
      "securities": [
        "AGG",
        "VTI",
```

NAV


```
            "SHY"
          ],
          "weights": [
            0.3345,
            0.6355,
            0.03
          ],
          "ret": 0.0413,
          "risk": 0.09
        }
    }
```


RESPONSE

Field	Description
`frontier`	Returned if `tgt_type` is excluded from the request. This includes the securities, weights, risk and return for the series of optimal portfolios.
`target_portfolio`	Returned if `tgt_type = risk` or `return`. This includes the securities, weights, risk and return for the given portfolio.
`securities`	The securities in the portfolio.
`weights`	The weights for each of the `securities` in the portfolio.
`ret`	The return of the portfolio.
`risk`	The risk of the portfolio.

NUCLEUS DATA DEPENDENCIES

- `Security`
- `Security Price`

Portfolio Optimization Score

This service assesses the relative health of a portfolio, as compared to a mean-variance optimized portfolio. The comparison is done on the basis of return-to-risk ratio, producing a score on a 0 to 100 scale, in which

NAV

100 represents the optimized portfolio. A portfolio optimization score defines how far a portfolio is away from the mean-variance efficient frontier in order to identify any improvement potential.

HTTP REQUEST

```
POST /portfolio_optimization_score
```

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "portfolio_tickers": ["AGG", "AAPL", "CHTR", "PFE", "NVDA"],
        "portfolio_weights": [0.1, 0.1, 0.1, 0.6, 0.1],
        "opt_config": {
            "tickers": ["AGG", "AAPL", "CHTR", "PFE", "NVDA"],
            "min_assets": 5,
            "w_asset_config": {
                "US_Equities": 1.0,
                "Fixed_Income": 1.0
            },
            "w_config": {
                "w_min_major": 0.05,
                "w_max_major": 1.0,
                "w_min_minor": 0.05,
                "w_max_minor": 1.0,
                "cash_amount": 0.0
            },
            "sec_types": ["minor", "minor", "minor", "minor", "minor"],
            "start_date": "2016-01-01",
            "end_date": "2017-01-01"
        },
        "use_proxy_data": false
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/portfolio_optimization_s
```

ARGUMENTS

Parameter	Description
`portfolio_tickers` *array, required*	Portfolio tickers, referencing securities defined in the Nucleus API.

NAV

Parameter	Description
`portfolio_weights` *array, required*	Portfolio weights, corresponding to `portfolio_tickers`. Must sum to `1.0`.
`tgt_type` *string*	The type of optimization to perform. Value may be `risk` or `return`. A `risk` optimization targets the risk of the current portfolio and attempts to maximize the return, while a `return` optimization targets the return of the current portfolio and attempts to minimize the risk. Defaults to `return`.
`opt_config` *map*	Settings for portfolio optimization. Includes the fields shown below.
`tickers` *array*	Security tickers to consider during the optimization process. Defaults to `portfolio_tickers`.
`min_assets` *integer*	Minimum number of assets included in the optimized portfolio. Defaults to `1`.
`w_asset_config` *map*	Weight constraints for asset classes. Stipulates a maximum weight for the asset classes represented in `tickers`. If an asset class is represented in `tickers` but not found in `w_asset_config`, the weight for that asset class will not be constrained.
`w_config` *map*	Weight constraints for security types, includes the fields shown below.
`w_min_major` *float*	Minimum weight for “major” securities. Defaults to `0.05`.
`w_max_major` *float*	Maximum weight for “major” securities. Defaults to `1`.
`w_min_minor` *float*	Minimum weight for “minor” securities. Defaults to `0.05`.
`w_max_minor` *float*	Maximum weight for “minor” securities. Defaults to `1`.
`cash_amount` *float*	Constant weight for “cash” securities. Defaults to `0`.
`sec_types` *array*	Security types, corresponding to tickers. May be `major`, `minor`, or `cash`. Defaults to `minor` for all `tickers`.

NAV

Parameter	Description
`start_date` *date*	Start date used for ticker price history. Defaults to earliest common date among `tickers` prices.
`end_date` *date*	End date used for ticker price history. Defaults to latest common date among `tickers` prices.
`use_proxy_data` *boolean*	If `true`, incorporate proxy price data as defined at the Security level in the Nucleus API. Proxy data is merged with base security data to form a continuous price history. Defaults to `false`.
`market_data_source` *string*	Source of security price data to be used in the analysis. Value may be `nucleus` or `integration`. When value is `nucleus`, security price data is sourced from Nucleus Security Price. When value is `integration`, security price data is sourced from a configured Integration Market Data vendor. Defaults to `nucleus`.
`account_id` *UUID, conditional requirement*	The ID of a Nucleus account used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`aggregation_account_id` *UUID, conditional requirement*	The ID of a Nucleus aggregation account used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`allocation_id` *UUID, conditional requirement*	The ID of a Nucleus allocation used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`model_id` *UUID, conditional requirement*	The ID of a Nucleus model used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`portfolio_id` *UUID, conditional requirement*	The ID of a Nucleus portfolio used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.

NAV

Parameter	Description
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

Example Response


```
{
    "current_portfolio": {
        "tickers": [
            "AGG",
            "AAPL",
            "CHTR",
            "PFE",
            "NVDA"
        ],
        "weights": [
            0.1,
            0.1,
            0.1,
            0.6,
            0.1
        ],
        "return": 0.2413,
        "risk": 0.1465
    },
    "optimized_portfolio": {
        "tickers": [
            "AAPL",
            "CHTR",
            "PFE",
            "NVDA",
            "AGG"
        ],
        "weights": [
            0.05,
            0.05,
            0.05,
            0.0745,
            0.7755
```

NAV


```
            ],
            "return": 0.2413,
            "risk": 0.0534
        },
        "optimization_score": 36.0
    }
```


RESPONSE

Field	Description
`current_portfolio`	Details on the current portfolio. Includes the fields shown below.
`tickers`	Tickers in the portfolio.
`weights`	Weights in the current portfolio, corresponding to `tickers`.
`return`	The annualized mean return.
`risk`	The annualized standard deviation of returns.
`optimized_portfolio`	Details on the optimized portfolio. Includes the fields shown below.
`tickers`	Tickers in the portfolio.
`weights`	Weights in the current portfolio, corresponding to `tickers`.
`return`	The annualized mean return.
`risk`	The annualized standard deviation of returns.
`optimization_score`	A score from 0 to 100 indicating the relative health of the current portfolio, 100 being the most optimal.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Security`
- `Security Price`
- `Account`
- `Portfolio`
- `Portfolio Holding`

NAV

- Model
- Model Holding
- Allocation
- Allocation Composition
- Aggregation Account
- Aggregation Account Holding

Only one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, and `portfolio_id` should be provided. An error will be thrown if multiple Nucleus data entities are referenced in a single request.

Portfolio Management

Rebalancing Signal

This service generates trade signals to strategically rebalance a model portfolio, with support for three types of rebalancing: period-based, drift-based, and Downside Protection. Period-based rebalancing generates a signal at the beginning of each time period with a pre-defined frequency. Drift-based rebalancing generates a signal whenever a portfolio holding deviates from its strategic target by a pre-defined amount. Downside Protection generates signals based on the observed risk of portfolio holdings, reducing exposure in stages as risk increases and restoring exposure in stages as risk decreases.

HTTP REQUEST

`POST /rebalancing_signal`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "model_id": "a83fd921-11ab-4003-a340-9bca9d3e01bd",
        "start_date": "2016-12-01",
        "end_date": "2017-12-08",
        "initial_weights": {
            "CMCSA": 0.05,
            "^IRX": 0.05,
            "KO": 0.05,
```

NAV

```
            "AMZN": 0.05,
            "GOOGL": 0.1,
            "CSCO": 0.1,
            "AGG": 0.1,
            "XOM": 0.1,
            "INTC": 0.1,
            "KHC": 0.1,
            "MCD": 0.1,
            "WMT": 0.1
        }
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/rebalancing_signal"
```

ARGUMENTS

Parameter	Description
`model_id` *UUID, required*	Identifier of the model in the Nucleus API to rebalance. In order to perform rebalancing, `downside`, `drift_rebal` or `period_rebal` must be set in the Nucleus endpoint `/model`. The `drift_factor` from `/model_holding` is used as of the `start_date` throughout the full simulation. If no drift factor is set in `/model_holding`, the `default_drift_factor` from `/model` is used. If `period_rebal`, set `rebalance_period`. If `drift_rebal`, set `default_drift_factor`. If `downside`, set `safe_sec` and `cash_sec`.
`start_date` *date, required*	Start date for analysis, represented in yyyy-mm-dd format
`end_date` *date, required*	End date for analysis, represented in yyyy-mm-dd format
`initial_weights` *map*	Initial weights for model holdings. If excluded, defaults to the model's `strategic_weight` from `/model_holding` as of the `start_date`.

Example Response

```
{
    "rebalance_signals": [
        {
            "date": "2017-03-28T00:00:00",
            "ticker": "CMCSA",
            "signal": "SELL",
```

NAV III


```
            "amount": 12.745833518909741,
            "type": 2
        },
        {
            "date": "2017-03-28T00:00:00",
            "ticker": "^IRX",
            "signal": "BUY",
            "amount": 155.94057228998332,
            "type": 2
        },
        {
            "date": "2017-03-28T00:00:00",
            "ticker": "KO",
            "signal": "SELL",
            "amount": 12.63570207140365,
            "type": 2
        },
        {
            "date": "2017-03-28T00:00:00",
            "ticker": "AMZN",
            "signal": "SELL",
            "amount": 16.0155517951892,
            "type": 2
        },
        {
            "date": "2017-03-28T00:00:00",
            "ticker": "GOOGL",
            "signal": "SELL",
            "amount": 18.65995935855528,
            "type": 2
        },
        {
            "date": "2017-03-28T00:00:00",
            "ticker": "CSCO",
            "signal": "SELL",
            "amount": 15.988445088739898,
            "type": 2
        },
        {
            "date": "2017-03-28T00:00:00",
            "ticker": "AGG",
            "signal": "SELL",
            "amount": 45.96572996834058,
            "type": 2
        },
        {
```

NAV ☰


```
            "date": "2017-03-28T00:00:00",
            "ticker": "XOM",
            "signal": "SELL",
            "amount": 3.144146724151982,
            "type": 2
        },
        {
            "date": "2017-03-28T00:00:00",
            "ticker": "INTC",
            "signal": "SELL",
            "amount": 5.25300352775003,
            "type": 2
        },
        {
            "date": "2017-03-28T00:00:00",
            "ticker": "MCD",
            "signal": "SELL",
            "amount": 12.879881718355065,
            "type": 2
        },
        {
            "date": "2017-03-28T00:00:00",
            "ticker": "WMT",
            "signal": "SELL",
            "amount": 12.65231851858788,
            "type": 2
        }
    ]
}
```


RESPONSE

Field	Description
`rebalance_signals`	The rebalancing signals for the specified model.
`date`	The date of the given rebalancing signal.
`ticker`	The unique identifier of a given security.
`signal`	The trade action, either `BUY` or `SELL`.
`amount`	The amount of the trade as a percentage. For example, `amount = 0.02`

NAV

Field	Description
	translates to selling 2% of a given `ticker`.
`type`	The type of trade signal. `1` = period-based, `2` = drift-based, and `4` = downside protection.

NUCLEUS DATA DEPENDENCIES

- `Model`
- `Model Holding`
- `Security`
- `Security Price`

Order Rebalance

Create order records necessary to rebalance an account or portfolio to its target. For an account, this endpoint will create order records necessary to rebalance all its portfolios according to the allocation(s) to which the account subscribes and models to which the portfolios subscribe. For a portfolio, this endpoint will create order records necessary to rebalance it to the model to which it subscribes. This rebalancing will generate orders for the buying of securities, selling of securities, or both.

The endpoint returns a series of `order_id` records and their underlying information, similar to the `GET /order` endpoint. All of the `order_ids` will be part of a bulk order and will have a `bulk_order_id` assigned to them. Note that this endpoint requires any corresponding portfolio(s) and model(s) to have asset size and holding records.

HTTP REQUEST

`POST /order_rebalance`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "portfolio_id": "ef93ce1c-b50e-4856-803a-db5332be93b0",
        "order_scope": "all",
        "buy_transaction_code_id": "f5af397b-7d22-433f-b01e-8202184a6386",
        "sell_transaction_code_id": "7d8d41d0-ed4b-4ae2-acb3-e0baed2ff1cc",
```

NAV

```
  "buy_threshold": 100,
  "cash_sec_id": "5276c9a5-5263-4aea-8027-327003738cef",
  "commit_orders": false,
  "non_fractional": true,
  "port_threshold": 100,
  "restrictions_on": false,
  "sell_threshold": 100,
  "use_cash_available": true,
  "use_strategic": false
}' "https://[sandbox][api].hydrogenplatform.com/proton/v1/order_rebalance"
```

ARGUMENTS

Parameter	Description
`portfolio_id` *UUID, conditional requirement*	The ID of the Nucleus Portfolio to rebalance. One of `portfolio_id` or `account_id` is required.
`account_id` *UUID, conditional requirement*	The ID of the Nucleus Account to rebalance. One of `portfolio_id` or `account_id` is required.
`order_scope` *string, required*	Value may be `all`, `buy_only`, or `sell_only`.
`buy_transaction_code_id` *UUID, conditional requirement*	The ID of the transaction code to denote a buy transaction. Conditionally required when `order_scope = all` or `buy_only`.
`sell_transaction_code_id` *UUID, conditional requirement*	The ID of the transaction code to denote a sell transaction. Conditionally required when `order_scope = all` or `sell_only`.
`buy_threshold` *float*	Buying threshold for the account as the number of shares. Defaults to `0`. Only applicable when `order_scope = all` or `buy_only`.
`cash_portfolio_id` *UUID*	The ID of the cash portfolio for the account. Only applicable when an `account_id` is provided and `order_scope = all` or `buy_only`.
`cash_sec_id` *UUID*	The ID of the cash security for the portfolio. Only applicable when a `portfolio_id` is provided and `order_scope = all` or `buy_only`.
`commit_orders` *boolean*	Indicates if the orders should be committed for execution. Defaults to `true` which indicates they should be committed.

NAV

Parameter	Description
`non_fractional` *boolean*	Indicates if purchasing/selling fractional shares of securities is allowed. Defaults to `false` which indicates it is allowed.
`port_threshold` *float*	Threshold for the minimum asset size of the portfolio as a monetary amount for rebalancing to take place. Defaults to `0`.
`restrictions_on` *boolean*	Indicates if there are restrictions on the account that should be followed. Defaults to `false` which indicates there are not.
`sell_threshold` *float*	Selling threshold for the account as the number of shares. Defaults to `0`.
`use_cash_available` *boolean*	Indicates if only the cash in the cash portfolio should be used. Defaults to `false` which indicates it should not. Only applicable when `order_scope = all` or `buy_only`.
`use_strategic` *boolean*	Indicates if the account should be synced to strategic weights as opposed to current weights. Defaults to `false` which indicates it should not.

Example Response


```
[
    {
        "id": "2fae86f5-a2ab-4d11-be1d-4ce08f903135",
        "create_date": "2018-02-07T19:29:37.000+0000",
        "transaction_code_id": "7d8d41d0-ed4b-4ae2-acb3-e0baed2ff1cc",
        "date": "2018-02-07",
        "price": 51.25,
        "is_read": true,
        "quantity": 26,
        "security_id": "52654829-e80b-4f16-9f8b-c9fde8a1aa63",
        "portfolio_id": "ef93ce1c-b50e-4856-803a-db5332be93b0",
        "account_id": "cce4cb18-9a0a-499e-be38-cfb089ba1781",
        "model_id": "ca74f042-eb32-4965-84c6-b0754d2115c5",
        "metadata": {}
    }
]
```

NAV

RESPONSE

Field	Description
`id`	The ID of the order.
`create_date`	The create date of the order.
`transaction_code_id`	The ID of the transaction code associated with the order.
`date`	The date of the order.
`price`	The price of the security associated with the order.
`is_read`	Indicates whether the order has been read. A value of `true` indicates that it has been read.
`quantity`	The quantity of securities associated with the order.
`security_id`	The ID of the security associated with the order.
`account_id`	The ID of the account associated with the order.
`portfolio_id`	The ID of the portfolio associated with the order.
`model_id`	The ID of the model associated with the order.
`metadata`	The metadata associated with the order.

NUCLEUS DATA DEPENDENCIES

- Account
- Account Allocation
- Allocation
- Model
- Model Holding
- Portfolio
- Portfolio Holding
- Security
- Security Price
- Transaction Code

NAV

Risk Scoring

Risk Score

This service calculates a risk score based on numerical answers to questions that evaluate risk appetite. The returned score is normalized on a scale from 0 to 100, where 0 indicates minimum risk appetite and 100 indicates maximum risk appetite.

HTTP REQUEST

`POST /risk_score`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "answers": [10, 400, 3, 9],
        "max_answers": [10, 1000, 4, 10],
        "weights": [0.25, 0.25, 0.25, 0.25]
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/risk_score"
```

ARGUMENTS

Parameter	Description
`answers` *array, conditional requirement*	Answer values. Values must be greater than or equal to zero. Higher answer values are assumed to indicate more appetite for risk.
`max_answers` *array, required*	Maximum possible answer values. Values must be greater than or equal to zero. Scales for `answers` and `max_answers` need not be consistent, as values are automatically normalized.
`weights` *array*	Weights for each answer. If sum is not 1, values are normalized proportionally. Example: `[60, 80, 60]` would be normalized to `[0.3, 0.4, 0.3]`. If excluded, answer values are equal-weighted.
`client_id` *UUID, conditional*	The ID of the Nucleus client used to derive `answers` and `weights`. If required parameter `answers` is not provided, `client_id` and `questionnaire_id` are

NAV

Parameter	Description
requirement	both required. If raw values for optional parameter `weights` are not provided, we will use `client_id` and `questionnaire_id` to try to derive these values instead.
`questionnaire_id` *UUID, conditional requirement*	The ID of a Nucleus questionnaire used to derive `answers` and `weights`. If required parameter `answers` is not provided, `client_id` and `questionnaire_id` are both required. If raw values for optional parameter `weights` are not provided, we will use `client_id` and `questionnaire_id` to try to derive these values instead.
`post_score` *boolean*	If `true`, a record of this client's risk score will be stored in Nucleus as a score, and the resulting `score_id` will be returned in the Proton response. A `client_id` must be provided to post a score. Defaults to `false`.

Example Response

```
{
    "risk_score": 76
}
```

RESPONSE

Field	Description
`risk_score`	A risk score from 0 to 100.
`score_id`	The unique id of the score record created in Nucleus. Only returned when `post_score = true` and a `client_id` is provided.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Client Response`
- `Questionnaire`

NAV

Dimensional Risk Score

This service calculates a risk score based on numerical answers to questions that evaluate risk appetite across multiple dimensions. Multi-dimensional analysis offers a more robust way to assess risk, as responses can be both strategically weighted and mapped to one or more distinct risk dimensions that are also strategically weighted. The returned score is normalized on a scale from 0 to 100, where 0 indicates minimum risk appetite and 100 indicates maximum risk appetite.

HTTP REQUEST

`POST /dimensional_risk_score`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "answers": [10, 400, 3, 9],
        "max_answers": [10, 1000, 4, 10],
        "answer_dims": [["dim1"], ["dim2"], ["dim3"], ["dim4"]],
        "dims": ["dim1", "dim2", "dim3", "dim4"],
        "dim_weights": [0.25, 0.25, 0.25, 0.25],
        "answer_weights": [0.25, 0.25, 0.25, 0.25]
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/dimensional_risk_score
```

ARGUMENTS

Parameter	Description
`answers` *array, conditional requirement*	Answer values. Values must be greater than or equal to zero. Higher answer values are assumed to indicate more appetite for risk.
`max_answers` *array, required*	Maximum possible answer values. Values must be greater than or equal to zero. Scales for `answers` and `max_answers` need not be consistent, as values are automatically normalized.
`answer_dims` *array, required*	Question risk dimensions. Maps individual answers to one or multiple risk dimensions. Should only include labels found in `dims`.

NAV

Parameter	Description
`dims` *array, required*	Labels for available risk dimensions. Each item should be a string and can be custom configured to reflect any number of dimensions. When more than one question maps to a given dimension, relationships between various `answer_weights` serve as the basis for weighting answers within that dimension.
`dim_weights` *array*	Weights for each risk dimension. Values correspond to and should match the length of `dims`. If sum is not 1, values are normalized proportionally. Example: `[60, 80, 60]` would be normalized to `[0.3, 0.4, 0.3]`. If excluded, dimensions are equal-weighted.
`answer_weights` *array*	Weights for each risk dimension answer. If sum is not 1, values are normalized proportionally. Example: `[60, 80, 60]` would be normalized to `[0.3, 0.4, 0.3]`. If excluded, answer values are equal-weighted.
`client_id` *UUID, conditional requirement*	The ID of the Nucleus client used to derive `answers` and `answer_weights`. If required parameter `answers` is not provided, `client_id` and `questionnaire_id` are both required. If raw values for optional parameter `answer_weights` are not provided, we will use `client_id` and `questionnaire_id` to try to derive these values instead.
`questionnaire_id` *UUID, conditional requirement*	The ID of a Nucleus questionnaire used to derive `answers` and `answer_weights`. If required parameter `answers` is not provided, `client_id` and `questionnaire_id` are both required. If raw values for optional parameter `answer_weights` are not provided, we will use `client_id` and `questionnaire_id` to try to derive these values instead.
`post_score` *boolean*	If `true`, a record of this client's risk score will be stored in Nucleus as a score, and the resulting `score_id` will be returned in the Proton response. A `client_id` must be provided to post a score. Defaults to `false`.

Example Response

```
{
    "risk_score": 76,
    "dim_scores": {
        "dim1": 100,
        "dim2": 40,
        "dim3": 75,
```

NAV

```
            "dim4": 90
        }
    }
```

RESPONSE

Field	Description
`risk_score`	A multi-dimensional risk score from 0 to 100.
`dim_scores`	A score from 0 to 100 for each dimension that was evaluated.
`score_id`	The unique id of the score record created in Nucleus. Only returned when `post_score = true` and a `client_id` is provided.

NUCLEUS DATA DEPENDENCIES

- `Client`
- `Client Response`
- `Questionnaire`

Risk Allocation

This service returns a portfolio based on an investor's risk score, drawing from a universe of pre-defined `allocations` or a dynamically generated efficient frontier. The risk score is used to conduct a percentile analysis within the portfolio universe; a risk score of 0 maps to the lowest risk portfolio within the universe, while a risk score of 100 maps to the highest risk portfolio within the universe. The user is matched to the most suitable allocation, given its standard deviation and the user's risk score.

HTTP REQUEST

`POST /risk_allocation`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
```

NAV

```
        "risk_score": 49,
        "allocation_method": "create",
        "opt_config":
         {
            "tickers": ["VTI", "AGG", "EFA", "VWO", "GLD", "SHY"],
            "min_assets": 1,
            "w_config": {
              "w_min_major": 0.05,
              "w_max_major": 1,
              "w_min_minor": 0.05,
              "w_max_minor": 1,
              "cash_amount": 0.03
            },
            "w_asset_config": {
              "US_Equities": 1.0,
              "Fixed_Income": 1.0,
              "Intl_Equities": 1.0,
              "EM_Equities": 1.0,
              "Commodities": 1.0
            },
            "sec_types": ["major", "major", "minor", "minor", "minor", "Cash"],
            "start_date": "2016-01-01",
            "end_date": "2017-01-01"
         },
        "allocations": ["347b1a30-342f-41f0-b484-78c350fc4243", "7688625f-ca41-47bd-a4
        "use_proxy_data": false
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/risk_allocation"
```

ARGUMENTS

Parameter	Description
`risk_score` *integer*	A risk score from 0 to 100. Defaults to 50 if not provided and not derived via `client_id`.
`allocation_method` *string, required*	The allocation universe resource. Value may be either `create` or `select`. `create` derives an allocation universe from a dynamically generated efficient frontier, based on `opt_config`. `select` derives an allocation universe from pre-defined allocations, as stipulated by the identifiers found in `allocations`.
`opt_config` *map, conditional requirement*	Portfolio optimization configuration, required if `allocation_method = create`. Information in `opt_config` refers to security data, which must be created in

NAV

Parameter	Description
	advance using the Nucleus endpoint `POST /security`. `opt_config` includes the following parameters:
`tickers` *array, required*	A list of symbols for securities included in the portfolio. We recommend using no more than 25 tickers in a single optimization. Pre-screening a securities universe prior to calling the service will result in faster execution times as well as more meaningful results.
`min_assets` *integer, required*	The minimum number of portfolio assets, excluding the cash security (if applicable).
`w_config` *map, required*	Weight constraints for security types. Values that can be set are `w_min_major`, `w_max_major`, `w_min_minor`, `w_max_minor`, and `cash_amount`, which stipulate lower and upper bounds for minor and major securities, as well as a constant weight for cash securities. Minimums are joined with zero to form a binary constraint. This means that the minimum weight of securities of each type may either be 0 (i.e. not included in the final portfolio) or the stipulated minimum. For example, a `w_min_major` constraint of 5% indicates that if a “major” security is chosen during the optimization, its weight must be greater than or equal to 5%. If, however, the security is not chosen, its weight is also allowable at 0%.
`w_asset_config` *map*	Weight constraints for asset classes. Stipulates a maximum weight for each asset class represented in tickers. If an asset class is not found, the max constraint defaults to 100%.
`sec_types` *array, required*	A type for each security. Values may be `major`, `minor`, and `cash`. `major` and `minor` designations interact with `w_config` to set distinct weight constraints for different kinds of securities. A maximum of one security may be defined as `cash`.
`start_date` *date*	Start date for historical prices, represented in yyyy-mm-dd format.
`end_date` *date*	End date for historical prices, represented in yyyy-mm-dd format.
`allocations` *array[UUID]*	List of `allocation_id`s in the Nucleus API to select from. As a pre-requisite, must have values for the `performance` and `volatility` arguments within the chosen `allocation_id`. Defaults to include all available allocations.
`use_proxy_data` *boolean*	If `true`, incorporate proxy price data as defined at the Security level in the Nucleus API. Proxy data is merged with base security data to form a continuous

NAV

Parameter	Description
	price history. Defaults to `false` .
`market_data_source` *string*	Source of security price data to be used in the analysis. Value may be `nucleus` or `integration` . When value is `nucleus` , security price data is sourced from Nucleus Security Price. When value is `integration` , security price data is sourced from a configured Integration Market Data vendor. Defaults to `nucleus` .
`client_id` *UUID*	The ID of the Nucleus client used to derive `risk_score` if `risk_score` is not provided.

Example Response

```
{
    "securities": [
        "AGG",
        "VTI"
    ],
    "weights": [
        0.5336,
        0.4664
    ],
    "ret": 0.061,
    "risk": 0.0627
}
```

RESPONSE

Field	Description
`securities`	If `allocation_method = create` , the tickers of the securities that should be included in the allocation created.
`weights`	If `allocation_method = create` , the weights of each security that should be included in the allocation created.
`id`	If `allocation_method = select` , the id of the allocation assigned to the client.
`ret`	The annualized return of the portfolio.

NAV

Field	Description
`risk`	The annualized standard deviation of the portfolio.

NUCLEUS DATA DEPENDENCIES

- `Security` ; required if `allocation_method = create`
- `Security Price` ; required if `allocation_method = create`
- `Allocation` ; `performance` and `volatility` must be set when `allocation_method = select`
- `Allocation Composition`
- `Model`
- `Model Holding`
- `Client`
- `Score`

Simulations

Event Study

When clients are weighing their options between portfolios, they can run a simulation to see how the portfolio would have performed during key historical events. This will help them understand a portfolio's exposure to various financial climates. The resulting performance and risk statistics can serve as a basis to evaluate a portfolio in the context of a given event.

The `events` parameter accepts the following event labels, each corresponding to a particular date range:

- `dot_com_bubble` : (5/14/2001 - 9/30/2002)
- `2008_financial_crisis` : (10/8/2007 - 03/02/2009)
- `brexit` : (6/23/2016 - 6/27/2016)
- `2011_black_monday` : (7/21/2011 - 8/10/2011)
- `september_eleventh` : (9/10/2001 - 9/21/2001)
- `1987_black_monday` : (10/5/1987 - 10/19/1987)
- `1992_black_wednesday` : (9/14/1992 - 10/9/1992)
- `1997_asian_financial_crisis` : (10/21/1997 - 10/27/1997)

HTTP REQUEST

```
POST /event_study
```

NAV ☰

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "portfolio_tickers": [
          "CVX",
          "PFE",
          "JNJ"
        ],
        "portfolio_weights": [
          0.3,
          0.4,
          0.3
        ],
        "events": [
          "2008_financial_crisis"
        ],
        "use_proxy_data": false
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/event_study"
```

ARGUMENTS

Parameter	Description
`portfolio_tickers` *array, conditional requirement*	Portfolio tickers, referencing securities defined in the Nucleus API.
`portfolio_weights` *array, conditional requirement*	Portfolio weights, corresponding to `portfolio_tickers`. Must sum to `1.0`.
`events` *array*	Historical events to analyze. Each item must be one of: `dot_com_bubble`, `2008_financial_crisis`, `brexit`, `2011_black_monday`, `september_eleventh`, `1987_black_monday`, `1992_black_wednesday`, `1997_asian_financial_crisis`. Defaults to all available events.
`use_proxy_data` *boolean*	If `true`, incorporate proxy price data as defined at the Security level in the Nucleus API. Proxy data is merged with base security data to form a continuous price history. Defaults to `false`.
`market_data_source` *string*	Source of security price data to be used in the analysis. Value may be `nucleus` or `integration`. When value is `nucleus`, security price data is

NAV

Parameter	Description
	sourced from Nucleus Security Price. When value is `integration`, security price data is sourced from a configured Integration Market Data vendor. Defaults to `nucleus`.
`account_id` *UUID, conditional requirement*	The ID of a Nucleus account used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`aggregation_account_id` *UUID, conditional requirement*	The ID of a Nucleus aggregation account used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`allocation_id` *UUID, conditional requirement*	The ID of a Nucleus allocation used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`model_id` *UUID, conditional requirement*	The ID of a Nucleus model used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`portfolio_id` *UUID, conditional requirement*	The ID of a Nucleus portfolio used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

Example Response

```
{
    "event_summary": [
        {
```

NAV


```
                "event_name": "2008_financial_crisis",
                "stats": {
                    "cumulative_return": -0.4068333255300416,
                    "annualized_return": -0.310936012869381,
                    "annualized_volatility": 0.3384477617390186
                }
            }
        ]
    }
```


RESPONSE

Field	Description
`event_summary`	Summary statistics for the events analyzed. Each underlying object contains the fields shown below.
`event_name`	Label for the event, as provided in `events`.
`stats`	Portfolio statistics for the duration of the event indicated by `event_name`.
`cumlative_return`	Cumulative return over the event duration.
`annualized_return`	Annualized return over the event duration.
`annualized_volatility`	Annualized standard deviation over the event duration.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Security`
- `Security Price`
- `Account`
- `Portfolio`
- `Portfolio Holding`
- `Model`

NAV

- Model Holding
- Allocation
- Allocation Composition
- Aggregation Account
- Aggregation Account Holding

This service depends upon Security Prices for the date ranges that correspond to `events` in order to run successfully. A given event will only be anlayzed if ALL Securities indicated by `portfolio_tickers`, `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` have the appropriate price data.

Only one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, and `portfolio_id` should be provided. An error will be thrown if multiple Nucleus data entities are referenced in a single request.

Model Backtest

This service simulates the behavior of a model over a specific date range in the past. It supports three types of model rebalancing: period-based, drift-based, and Downside Protection. Period-based rebalancing generates a signal at the beginning of each time period with pre-defined frequency. Drift-based rebalancing generates a signal whenever a model holding deviates from its strategic target by a pre-defined amount. Downside Protection generates signals based on the observed risk of model holdings, reducing exposure in stages as risk increases and restoring exposure in stages as risk decreases. These rebalancing types can be assigned to a model in the Nucleus API Model Service.

HTTP REQUEST

`POST /backtest`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "model_id": "9aad12e8-25dc-421d-a041-1b157fff7738",
        "start_date": "2016-12-01",
```

NAV

```
    "end_date": "2016-12-05",
    "initial_weights": {
      "GOOGL": 0.25,
      "CSCO": 0.25,
      "MCD": 0.25,
      "WMT": 0.25
    },
    "asset_size": 1000,
    "asset_sizes": true,
    "trades": true,
    "holdings": true,
    "stats": true
  }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/backtest"
```

ARGUMENTS

Parameter	Description
`model_id` *UUID, required*	Identifier of the model in the Nucleus API to backtest. In order to perform the backtest, `downside`, `drift_rebal` or `period_rebal` must be set in the Nucleus endpoint `/model`. The `drift_factor` from `/model_holding` is used as of the `start_date` throughout the full simulation. If no drift factor is set in `/model_holding`, the `default_drift_factor` from `/model` is used. If `period_rebal`, set `rebalance_period`. If `drift_rebal`, set `default_drift_factor`. If `downside`, set `safe_sec` and `cash_sec`.
`start_date` *date, required*	Start date for analysis, represented in yyyy-mm-dd format.
`end_date` *date, required*	End date for analysis, represented in yyyy-mm-dd format.
`initial_weights` *map*	Initial weights for model holdings. If excluded, defaults to the model's `strategic_weight` from `/model_holding` as of the `start_date`.
`asset_size` *float*	The initial asset size. If excluded, defaults to `1000`.
`asset_sizes` *boolean*	If `true`, show the asset sizes from the backtest. If excluded, defaults to `true`.
`trades` *boolean*	If `true`, show the trades from the backtest. If excluded, defaults to `true`.

NAV

Parameter	Description
`holdings` *boolean*	If `true`, show the holdings from the backtest. If excluded, defaults to `true`.
`stats` *boolean*	If `true`, show risk and return statistics from the backtest. If excluded, defaults to `true`.

Example Response


```
{
    "asset_sizes": {
        "2016-12-01T00:00:00": 1000,
        "2016-12-02T00:00:00": 998.6022625755537,
        "2016-12-05T00:00:00": 1004.3918279210744
    },
    "trades": [
        {
            "date": "2016-12-05T00:00:00",
            "ticker": "GOOGL",
            "signal": "SELL",
            "amount": 0.00039250387412376186,
            "type": 2
        },
        {
            "date": "2016-12-05T00:00:00",
            "ticker": "CSCO",
            "signal": "BUY",
            "amount": 0.0013502458022483788,
            "type": 2
        },
        {
            "date": "2016-12-05T00:00:00",
            "ticker": "MCD",
            "signal": "BUY",
            "amount": 0.0001361108332734451,
            "type": 2
        },
        {
            "date": "2016-12-05T00:00:00",
            "ticker": "WMT",
            "signal": "SELL",
```

NAV


```
                "amount": 0.0010938527613980065,
                "type": 2
            }
        ],
        "holdings": [
            {
                "date": "2016-12-02T00:00:00",
                "tickers": [
                    "GOOGL",
                    "CSCO",
                    "MCD",
                    "WMT"
                ],
                "weights": [
                    0.25039250387412376,
                    0.24864975419775162,
                    0.24986388916672655,
                    0.251093852761398
                ],
                "amount": [
                    250.04252090065808,
                    248.30220713073004,
                    249.51464505782053,
                    250.742889486345
                ]
            },
            {
                "date": "2016-12-05T00:00:00",
                "tickers": [
                    "GOOGL",
                    "CSCO",
                    "MCD",
                    "WMT"
                ],
                "weights": [
                    0.2530329073979014,
                    0.2509383051693015,
                    0.2507662014695168,
                    0.24526258596328018
                ],
                "amount": [
                    254.14418438556214,
                    252.0403830244111,
                    251.8675234747924,
                    246.33973703630863
                ]
```

NAV


```
            }
        ],
        "stats": {
            "cumulative_return": 0.004391827921074398,
            "annualized_return": 0.7331968693633459,
            "annualized_volatility": 0.08076823639707757
        }
    }
```


RESPONSE

Field	Description
`asset_sizes`	The asset size per day, beginning on the `start_date`, for the given time period.
`trades`	Rebalancing signals for the given model.
`date`	The date of the signal.
`ticker`	The ticker of the security.
`signal`	The signal, either BUY or SELL.
`amount`	The amount of the trade.
`type`	The type of trade signal. `1` = period-based, `2` = drift-based, and `4` = downside protection.
`holdings`	The model's composition by date.
`date`	The date for the holding details.
`tickers`	The tickers of the holdings.
`weights`	The weights of the holdings, represented on a percentage basis.
`amount`	The total value for each security.
`stats`	Model metrics for the duration of the backtest.
`cumulative_return`	The cumulative return.

NAV

Field	Description
`annualized_return`	The annualized return.
`annualized_volatility`	The annualized standard deviation of returns.

NUCLEUS DATA DEPENDENCIES

- Model
- Model Holding
- Security
- Security Price

Monte Carlo

This service extrapolates the future value of a hypothetical investment account via a random walk Monte Carlo. Future values are approximated by simulating individual periods of portfolio growth based on samples derived from the given historical performance and volatility portfolio attributes, in conjunction with ongoing cash inflows or outflows (which occur at the end of each period) in the simulated account. The return samples are stochastic; they assume portfolio returns are distributed normally, and are chained together in a random walk sequence.

HTTP REQUEST

`POST /monte_carlo`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "cf": [[100,10]],
        "init_bal": 1000,
        "ret_mod": [0],
        "mu": [0.05],
        "sigma": [0.06],
        "n": 1000,
        "remove_outliers": false,
```

NAV

```
    "result_type": "median",
    "p": [50],
    "min_bal": [0],
    "max_bal": [4000],
    "min_sample": [0],
    "max_sample": [0.15]
}' "https://[sandbox][api].hydrogenplatform.com/proton/v1/monte_carlo"
```

ARGUMENTS

Parameter	Description
`cf` *array, required*	The cash flows to occur during the simulation. This parameter is an array containing sub-arrays with 2 elements. The first element of each sub-array is a cash flow amount, and the second is a time period duration. Example: `[[100.00,10], [250.00,15]]`. This example indicates cash flows of 100.00 will take place for 10 consecutive time periods, followed by cash flows of 250 for 15 periods (for a total of 25 time periods). Cash flows of 0 must be defined explicitly.
`mu` *array, conditional requirement*	The periodic mean portfolio return. The length of the array must be less than or equal to the number of time periods present in `cf`. If it is less, values will be conformed to the number of time periods. For example, `[0.04, 0.06]` would result in an implied `mu` of `[0.04, 0.04, 0.06, 0.06]` for a simulation that spans four time periods.
`sigma` *array, conditional requirement*	The periodic portfolio standard deviation. The length of the array must be less than or equal to the number of time periods present in `cf` If it is less, values will be conformed to the number of time periods. Values must be greater than 0.
`ret_mod` *array*	A periodic return modifier. Use this value to adjust the randomly generated return for each period of the simulation on an absolute basis. For example, to reflect a fee of 1% taken out each period, use `[-0.01]`. If excluded, defaults to `[0]`.
`init_bal` *float, conditional requirement*	The initial investment at time zero. If excluded, defaults to `100`.
`n` *integer*	The number of simulations to run. Larger values will increase the execution time of the service. The maximum acceptable value is `10000`. If excluded, defaults to `1000`.

NAV

Parameter	Description
`remove_outliers` *boolean*	If `true`, remove outlying results. If `true`, outlier analysis is performed on a median absolute deviation (MAD) basis, at the 2.5 threshold. If excluded, defaults to `false`.
`result_type` *string*	The type of simulation results to display in the output. Value may be `raw`, `mean`, `median`, `tens`, or `custom`. `raw` returns all simulation results; `mean` returns the average simulation result; `median` returns the median simulation result; `tens` return the 10th, 20th, 30th, 40th, 50th, 60th, 70th, 80th, and 90th percentile simulation results; `custom` returns results for the percentile(s) defined in `p`. For example, `[30, 50, 70]` would return the 30th, 50th, and 70th percentiles. If excluded, defaults to `raw`.
`p` *array*	Custom result percentile(s). If excluded, defaults to `[50]`.
`min_bal` *array*	A lower bound to enforce on the periodic balance. Each item in the array corresponds to a period in the simulation. If the length of the array is less than the number of simulation periods, it is conformed to the appropriate length by persisting the final value in the array.
`max_bal` *array*	An upper bound to enforce on the periodic balance. Each item in the array corresponds to a period in the simulation. If the length of the array is less than the number of simulation periods, it is conformed to the appropriate length by persisting the final value in the array.
`min_sample` *array*	A lower bound to enforce on the randomly sampled periodic return. Each item in the array corresponds to a period in the simulation. If the length of the array is less than the number of simulation periods, it is conformed to the appropriate length by persisting the final value in the array.
`max_sample` *array*	An upper bound to enforce on the randomly sampled periodic return. Each item in the array corresponds to a period in the simulation. If the length of the array is less than the number of simulation periods, it is conformed to the appropriate length by persisting the final value in the array.
`use_proxy_data` *boolean*	If `true`, incorporate proxy price data as defined at the Security level in the Nucleus API. Proxy data is merged with base security data to form a continuous price history. Defaults to `false`.
`market_data_source` *string*	Source of security price data to be used in the analysis. Value may be `nucleus` or `integration`. When value is `nucleus`, security price data is sourced from Nucleus Security Price. When value is `integration`,

NAV

Parameter	Description
	security price data is sourced from a configured Integration Market Data vendor. Defaults to `nucleus`.
`account_id` *UUID, conditional requirement*	The ID of a Nucleus account used to derive `mu`, `sigma`, and `init_bal`. If `mu` or `sigma` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`aggregation_account_id` *UUID, conditional requirement*	The ID of a Nucleus aggregation account used to derive `mu`, `sigma`, and `init_bal`. If `mu` or `sigma` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`allocation_id` *UUID, conditional requirement*	The ID of a Nucleus allocation used to derive `mu`, `sigma`, and `init_bal`. If `mu` or `sigma` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`model_id` *UUID, conditional requirement*	The ID of a Nucleus model used to derive `mu`, `sigma`, and `init_bal`. If `mu` or `sigma` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`portfolio_id` *UUID, conditional requirement*	The ID of a Nucleus portfolio used to derive `mu`, `sigma`, and `init_bal`. If `mu` or `sigma` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`frequency_interval` *string*	The periodicity to convert daily values to when deriving `mu` and `sigma` from Nucleus via an `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id`. When `mu` are `sigma` are provided in the raw request, this parameter has no impact. Must be one of `year`, `quarter`, `month`, `week`, or `day`. Defaults to `year`.
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

NAV

Example Response


```
{
    "sims": [
        [
            1000,
            1150.7407772024958,
            1323.0962376032248,
            1490.070327954984,
            1672.015451231377,
            1863.3772622510578,
            2066.324013845525,
            2275.2452616749324,
            2501.7737472360695,
            2744.753718748225,
            2993.112513070986
        ]
    ]
}
```


RESPONSE

Field	Description
`sims`	A collection of Monte Carlo results. If `result_type = raw`, this returns multiple sets of results, defined by `n`. If `result_type = custom`, this returns the results in the order defined by `p`. If `result_type = tens`, this returns the results in the order of 10, 20, 30, 40, 50, 60, 70, 80, 90. If `result_type = mean` or `median` only one array of results will be returned.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Security`
- `Security Price`
- `Account`
- `Portfolio`
- `Portfolio Holding`
- `Portfolio Asset Size`
- `Model`

NAV

- Model Holding
- Model Asset Size
- Allocation
- Allocation Composition
- Aggregation Account
- Aggregation Account Holding
- Aggregation Account Balance

Only one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, and `portfolio_id` should be provided. An error will be thrown if multiple Nucleus data entities are referenced in a single request.

Portfolio What-If

As clients' wants and needs change, it is useful to evaluate shifts in a portfolio's makeup. This tool considers the effect of "what-if" portfolio composition changes such as adding, removing, reducing, or increasing various positions in a portfolio. This can help a client make decisions around which securities to buy or sell, or help a wealth manager assess portfolio composition over time. The analysis between the current and altered portfolios is done on the basis annualized historical return and annualized historical standard deviation.

HTTP REQUEST

`POST /portfolio_what_if`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "current_portfolio_tickers": [
          "CVX",
          "PFE",
          "JNJ"
        ],
        "current_portfolio_weights": [
          0.20,
          0.20,
          0.60
```

NAV

```
    ],
    "altered_portfolio_tickers": [
      "CVX",
      "PFE",
      "JNJ"
    ],
    "altered_portfolio_weights": [
      0.30,
      0.30,
      0.40
    ],
    "start_date": "2017-01-01",
    "end_date": "2018-12-31",
    "use_proxy_data": false
  }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/portfolio_what_if"
```

ARGUMENTS

Parameter	Description
`current_portfolio_tickers` *array, conditional requirement*	Current portfolio tickers, referencing securities defined in the Nucleus API.
`current_portfolio_weights` *array, conditional requirement*	Current portfolio weights, corresponding to `current_portfolio_tickers`. Must sum to `1.0`.
`altered_portfolio_tickers` *array*	Altered portfolio tickers, referencing securities defined in the Nucleus API. Defaults to `current_portfolio_tickers`.
`altered_portfolio_weights` *array, required*	Altered portfolio weights, corresponding to `altered_portfolio_tickers`. Must sum to `1.0`.
`start_date` *date*	Start date used for ticker price history. Defaults to earliest common date among `current_portfolio_tickers` and `altered_portfolio_tickers` prices.
`end_date` *date*	End date used for ticker price history. Defaults to latest common date among `portfolio_tickers` and `altered_portfolio_tickers` prices.
`use_proxy_data` *boolean*	If `true`, incorporate proxy price data as defined at the Security level in the Nucleus API. Proxy data is merged with base security data to form a continuous price history. Defaults to `false`.

NAV

Parameter	Description
`market_data_source` *string*	Source of security price data to be used in the analysis. Value may be `nucleus` or `integration`. When value is `nucleus`, security price data is sourced from Nucleus Security Price. When value is `integration`, security price data is sourced from a configured Integration Market Data vendor. Defaults to `nucleus`.
`account_id` *UUID, conditional requirement*	The ID of a Nucleus account used to derive `current_portfolio_tickers` and `current_portfolio_weights`. If `current_portfolio_tickers` or `current_portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`aggregation_account_id` *UUID, conditional requirement*	The ID of a Nucleus aggregation account used to derive `current_portfolio_tickers` and `current_portfolio_weights`. If `current_portfolio_tickers` or `current_portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`allocation_id` *UUID, conditional requirement*	The ID of a Nucleus allocation used to derive `current_portfolio_tickers` and `current_portfolio_weights`. If `current_portfolio_tickers` or `current_portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`model_id` *UUID, conditional requirement*	The ID of a Nucleus model used to derive `current_portfolio_tickers` and `current_portfolio_weights`. If `current_portfolio_tickers` or `current_portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`portfolio_id` *UUID, conditional requirement*	The ID of a Nucleus portfolio used to derive `current_portfolio_tickers` and `current_portfolio_weights`. If `current_portfolio_tickers` or `current_portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

NAV

Example Response


```
{
    "annualized_return": {
        "current_portfolio": 0.158543872616828,
        "altered_portfolio": 0.13392921668177515,
        "delta": -0.024614655935052854
    },
    "annualized_volatility": {
        "current_portfolio": 0.005478420111416463,
        "altered_portfolio": 0.005193612691605431,
        "delta": -0.00028480741981103206
    }
}
```


RESPONSE

Field	Description
`annualized_return`	Annualized return information for the current and altered portfolios.
`current_portfolio`	Current portfolio's annualized return.
`altered_portfolio`	Altered portfolio's annualized return.
`delta`	Raw difference in annualized return between the current and altered portfolios.
`annualized_volatility`	Annualized portfolio standard deviation.
`current_portfolio`	Current portfolio's annualized standard deviation.
`altered_portfolio`	Altered portfolio's annualized standard deviation.
`delta`	Raw difference in annualized standard deviation between the current and altered portfolios.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when

NAV

Field	Description
	`create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Security`
- `Security Price`
- `Account`
- `Portfolio`
- `Portfolio Holding`
- `Model`
- `Model Holding`
- `Allocation`
- `Allocation Composition`
- `Aggregation Account`
- `Aggregation Account Holding`

Only one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, and `portfolio_id` should be provided. An error will be thrown if multiple Nucleus data entities are referenced in a single request.

Scenario Analysis

When assessing a portfolio, it is useful to simulate its response to a series of external factors. Scenario Analysis provides a statistical estimate of the impact that a combination of external movements may have on a portfolio's rate of return, based on a multivariate regression model that considers historical price behavior. In order to support many types of real-world situations, this tool accounts for the magnitude, direction, and duration of a given change (e.g. a 10% upward movement over a period of 2 weeks). It also supports the ability to reflect a lag between the movement of each factor and the portfolio (e.g. the portfolio is impacted 3 days after the factor movement). The resulting analysis can be helpful in comparing the economic exposures of different portfolios.

HTTP REQUEST

```
POST /scenario_analysis
```

Example Request

NAV

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "portfolio_tickers": [
          "HD",
          "PG",
          "EBAY",
          "NVDA"
        ],
        "portfolio_weights": [
          0.25,
          0.25,
          0.25,
          0.25
        ],
        "frequency_interval": "month",
        "scenario": [
          {
            "ticker": "VNQ",
            "change_amount": 0.02,
            "change_duration": 1,
            "lag": 0
          },
          {
            "ticker": "GLD",
            "change_amount": -0.05,
            "change_duration": 3,
            "lag": 1
          }
        ],
        "start_date": "2015-08-11",
        "end_date": "2016-12-31",
        "trading_days_per_year": 252,
        "use_proxy_data": false
      }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/scenario_analysis"
```

ARGUMENTS

Parameter	Description
`portfolio_tickers` *array, conditional requirement*	Portfolio tickers, referencing securities defined in the Nucleus API.

NAV

Parameter	Description
`portfolio_weights` *array, conditional requirement*	Portfolio weights, corresponding to `portfolio_tickers`. Must sum to `1.0`.
`frequency_interval` *float, required*	Unit of time associated with `change_duration` and `lag`. Must be one of `year`, `quarter`, `month`, `week`, or `day`.
`scenario` *map, required*	A scenario to analyze, defined as a series of hypothetical price movements for securities, indices, or other instruments.
`ticker` *string, required*	Ticker representing the factor, referencing a security defined in the Nucleus API.
`change_amount` *float, required*	A percentage price change, represented in decimal format. Value can be positive or negative.
`change_duration` *integer, required*	Number of time periods over which `change_amount` is to take place, where each period represents `frequency_interval`.
`lag` *integer, required*	Number of time periods between the factor signal and the portfolio's response to that signal, where each period represents `frequency_interval`. For example, a value of 3 indicates that the portfolio responds 3 periods after the factor moves. Value cannot be negative.
`start_date` *date*	Start date used for ticker price history. Defaults to earliest common date among `portfolio_tickers` and `scenario` ticker prices.
`end_date` *date*	End date used for ticker price history. Defaults to latest common date among `portfolio_tickers` and `scenario` ticker prices.
`trading_days_per_year` *integer*	The number of days per year for which a portfolio is subject to market fluctuation. Defaults to 252.
`use_proxy_data` *boolean*	If `true`, incorporate proxy price data as defined at the Security level in the Nucleus API. Proxy data is merged with base security data to form a continuous price history. Defaults to `false`.
`market_data_source` *string*	Source of security price data to be used in the analysis. Value may be `nucleus` or `integration`. When value is `nucleus`, security price data is sourced from Nucleus Security Price. When value is `integration`, security price data is sourced from a configured Integration Market Data vendor. Defaults to `nucleus`.

NAV

Parameter	Description
`account_id` *UUID, conditional requirement*	The ID of a Nucleus account used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`aggregation_account_id` *UUID, conditional requirement*	The ID of a Nucleus aggregation account used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`allocation_id` *UUID, conditional requirement*	The ID of a Nucleus allocation used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`model_id` *UUID, conditional requirement*	The ID of a Nucleus model used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`portfolio_id` *UUID, conditional requirement*	The ID of a Nucleus portfolio used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

Due to the relationship between `change_duration`, `lag`, `start_date`, and `end_date`, some input configurations may lead to an insufficient amount of price data. In these cases, the analysis will not run and you will receive a 400 status.

Example Response

NAV


```
{
    "portfolio_impact": 0.010304645392564492,
    "scenario_results": [
        {
            "ticker": "VNQ",
            "t_statistic": 4.770856786757572,
            "regression_coefficient": 0.8023214304210411
        },
        {
            "ticker": "GLD",
            "t_statistic": 0.6448259002724069,
            "regression_coefficient": 0.1148356643171266
        }
    ]
}
```


RESPONSE

Field	Description
`portfolio_impact`	Expected impact of `scenario` factors on the portfolio's return. The associated unit of time is the base unit indicated by `frequency_interval` in the request (for example, `week` = 1 week).
`scenario_results`	Underlying results for each `scenario` factor, including the fields shown below.
`ticker`	Ticker to indicate the factor.
`t_statistic`	Raw t-statistic from the regression analysis, typically used to determine statistical significance of `regression_coefficient`.
`regression_coefficient`	Statistical relationship between the `ticker` return and the portfolio return.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Security`
- `Security Price`
- `Account`

NAV

- Portfolio
- Portfolio Holding
- Model
- Model Holding
- Allocation
- Allocation Composition
- Aggregation Account
- Aggregation Account Holding

Only one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, and `portfolio_id` should be provided. An error will be thrown if multiple Nucleus data entities are referenced in a single request.

Sensitivity Analysis

When assessing a portfolio, simulating its response to an external factor can give greater insight into the portfolio's exposure. Sensitivity Analysis provides a statistical estimate of the impact that an external movement may have on a portfolio's rate of return, based on an ordinary-least-squares linear regression model that considers historical price behavior. In order to support many types of real-world situations, this tool accounts for the magnitude, direction, and duration of a given change (e.g. a 10% upward movement over a period of 2 weeks). It also supports the ability to reflect a lag between the movement of the factor and the portfolio (e.g. the portfolio is impacted 3 days after the factor movement). The resulting analysis can be helpful in comparing the economic and risk exposures of different portfolios.

HTTP REQUEST

`POST /sensitivity_analysis`

Example Request

```
curl -X POST -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d '{
        "portfolio_tickers": [
            "HD",
            "PG",
            "EBAY",
            "NVDA"
```

NAV

```
        ],
        "portfolio_weights": [
            0.25,
            0.25,
            0.25,
            0.25
        ],
        "frequency_interval": "day",
        "sensitivity_factor": {
            "ticker": "AGG",
             "change_amount": -0.05,
             "change_duration": 3,
             "lag": 0
        },
        "start_date": "2015-08-11",
        "end_date": "2016-12-31",
        "trading_days_per_year": 252,
        "use_proxy_data": false
    }' "https://[sandbox][api].hydrogenplatform.com/proton/v1/sensitivity_analysis"
```

ARGUMENTS

Parameter	Description
`portfolio_tickers` *array, conditional requirement*	Portfolio tickers, referencing securities defined in the Nucleus API.
`portfolio_weights` *array, conditional requirement*	Portfolio weights, corresponding to `portfolio_tickers`. Must sum to `1.0`.
`frequency_interval` *string, required*	Unit of time associated with `change_duration` and `lag`. Must be one of `year`, `quarter`, `month`, `week`, or `day`.
`sensitivity_factor` *map, required*	A factor to analyze, defined as a hypothetical price movement for a security, index, or other instrument.
`ticker` *string, required*	Ticker representing the factor, referencing a security defined in the Nucleus API.
`change_amount` *float, required*	A percentage price change, represented in decimal format. Can be positive or negative.

NAV

Parameter	Description
`change_duration` *integer, required*	Number of time periods over which `change_amount` is to take place, where each period represents `frequency_interval`.
`lag` *integer*	Number of time periods between the factor signal and the portfolio's response to that signal, where each period represents `frequency_interval`. For example, a value of 3 indicates that the portfolio responds 3 periods after the factor moves. Defaults to 0, and cannot be negative.
`start_date` *date*	Start date used for ticker price history. Defaults to earliest common date among `portfolio_tickers` and `sensitivity_factor ticker` prices.
`end_date` *date*	End date used for ticker price history. Defaults to latest common date among `portfolio_tickers` and `sensitivity_factor ticker` prices.
`trading_days_per_year` *integer*	The number of days per year for which a portfolio is subject to market fluctuation. Defaults to 252.
`use_proxy_data` *boolean*	If `true`, incorporate proxy price data as defined at the Security level in the Nucleus API. Proxy data is merged with base security data to form a continuous price history. Defaults to `false`.
`market_data_source` *string*	Source of security price data to be used in the analysis. Value may be `nucleus` or `integration`. When value is `nucleus`, security price data is sourced from Nucleus Security Price. When value is `integration`, security price data is sourced from a configured Integration Market Data vendor. Defaults to `nucleus`.
`account_id` *UUID, conditional requirement*	The ID of a Nucleus account used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`aggregation_account_id` *UUID, conditional requirement*	The ID of a Nucleus aggregation account used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`allocation_id` *UUID, conditional requirement*	The ID of a Nucleus allocation used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is

NAV

Parameter	Description
	not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`model_id` *UUID, conditional requirement*	The ID of a Nucleus model used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`portfolio_id` *UUID, conditional requirement*	The ID of a Nucleus portfolio used to derive `portfolio_tickers` and `portfolio_weights`. If `portfolio_tickers` or `portfolio_weights` is not provided, one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, or `portfolio_id` is required.
`create_log` *boolean*	If `true`, a record of this Proton request URL, request body, response body, and Nucleus UUID(s), where applicable, will be stored in Electron as an audit log, and the resulting `audit_log_id` will be returned in the Proton response. Defaults to `false`.

Due to the relationship between `change_duration`, `lag`, `start_date`, and `end_date`, some input configurations may lead to an insufficient amount of price data. In these cases, the analysis will not run and you will receive a 400 status.

Example Response

```
{
    "portfolio_impact": 0.03466700648259927,
    "t_statistic": -2.401636667959841,
    "margin_of_error": 0.02829157510133723
}
```

RESPONSE

Field	Description
`portfolio_impact`	Expected impact of `sensitivity_factor` on the portfolio's return. The associated unit of time is the base unit indicated by `frequency_interval` in the

NAV

Field	Description
	request (for example, `week` = 1 week).
`t_statistic`	Raw t-statistic from the regression analysis, typically used to determine statistical significance of the regression coefficient.
`margin_of_error`	Estimated margin of error pertaining to `portfolio_impact` at a 95% confidence level.
`audit_log_id`	The unique id of the audit log record created in Electron. Only returned when `create_log = true`.

NUCLEUS DATA DEPENDENCIES

- `Security`
- `Security Price`
- `Account`
- `Portfolio`
- `Portfolio Holding`
- `Model`
- `Model Holding`
- `Allocation`
- `Allocation Composition`
- `Aggregation Account`
- `Aggregation Account Holding`

Only one of `account_id`, `aggregation_account_id`, `allocation_id`, `model_id`, and `portfolio_id` should be provided. An error will be thrown if multiple Nucleus data entities are referenced in a single request.

Appendix

This appendix contains additional information to elaborate on the underlying algorithm methodology.

NAV III

Financial Health

Financial Health Check

Applicable Proton Services

- `POST /financial_health_check`

Methodology

The Financial Ratio Calculation results are compared to the Ideal Ratio. If the Financial Ratio is the Ideal Ratio Result or higher, the Ratio Test Result is marked as Pass. If the Financial Ratio is outside of the Ideal Ratio Result or lower, the Ratio Test Result is marked as Fail.

The Percentile Scale is calculated in the following way:

- *If a Calculation Result is the Ideal Result or higher, the Percentile Scale returns 100*
- *If a Calculation Result is less than the the Ideal Result, the Percentile Scale returns = "Calculation Result" / "Ideal Result"*

Notes & Considerations

Ideal Results & Ideal Direction of Outputs:

1. `liquidity_ratio_expenses` : x >= 250% , Higher
2. `liquidity_ratio_liabilities` : x >= 10%, Higher
3. `current_ratio` : x >= 50% , Higher
4. `savings_ratio_gross` : x >= 10% , Higher
5. `savings_ratio_net` : x >= 10% , Higher
6. `asset_allocation_ratio` : x >= 15% , Higher

References

i. https://pdfs.semanticscholar.org/269d/5f90e8ce1951d74c61e2a3f8ed116c4eb021.pdf
ii. https://www.afcpe.org/assets/journals/vol_13.pdf

Goals

Goals Framework

- Goals are described in terms of net purchasing power. This means that certain variables, including monetary inflation and taxes paid on the redemption of capital, are accounted for explicitly. For example, a target goal of 100 may correspond to an implied outcome of 125 if the capital is assumed to be taxed at a rate of 20% upon redemption. The purpose of this mechanism is to better reflect real-world purchasing power through the culmination of the goal horizon.
- All goals are classified as either accumulation or decumulation. Accumulation goals are defined by a target amount of capital in the future, assumed to be redeemed in full upon termination of the goal horizon. Decumulation goals are defined by redeeming a target amount of capital periodically over a horizon, which may also include an accumulation phase.
- Configurations that involve contributions of capital (via an initial investment sum or an ongoing recurring deposit, for example) will prioritize a minimization of the present value of cash outlays. This built-in engine assumes that goal achievement with minimal cash expenditure is most desirable, ceteris paribus.
- Goal achievement probabilities are extrapolated via Monte Carlo analysis. For more information on the Monte Carlo methodology, refer to the Monte Carlo Appendix section of this appendix. The distribution of outcomes generated by this process allows for an assessment of the likelihood that a goal will be attained. Please note that such goal probabilities are uncertain by nature, and repeating this type of simulation-based analysis may not yield precisely the same results.
- Simulation results generated in the goals framework are returned on a yearly basis. The first item always indicates the investment value at t0, such that each simulation result for a 10-year goal horizon will contain 11 observations (t0 plus one observation for each year in the goal horizon).
- Goal achievement indicators are derived from the relationship between goal achievement probability and a desired confidence target. Confidence targets serve as the basis for a percentile analysis of Monte Carlo outcomes, in which outcomes below the ((1 – c) * 100)th percentile, where c is the confidence target on a scale from 0 to 1, are excluded as a left-tail margin. If the confidence target is not satisfied, this is reflected by a goal probability that is less than the confidence target.
- Because individual outcomes are binary in nature, the framework supports optional goal deviation thresholds in order to promote robustness. Thresholds may be set on an absolute basis or a percentage basis, and serve as an acceptable error margin on the nominal goal amount. For example, a goal of 10,000 modified with a 0.50% acceptable deviation threshold indicates that outcomes as low as 9,950 will be considered affirmative for goal achievement.
- For decumulation goals, deposits are assumed to stop at the end of the accumulation horizon.

Goal Allocation Appendix

Applicable Proton Services

- `POST /goal_accumulation/allocation`

NAV

- `POST /goal_decumulation/allocation`

Methodology

For the “risk” allocation_priority, the chosen portfolio is based on the defined risk_score (see Risk Allocation for more information). For the “goal” allocation_priority, the engine searches through a series of potential goal plan configurations in which the portfolio risk and return change while the initial investment amount, periodic deposit amount, goal horizon, and target goal amount are held constant. The lowest risk portfolio that projects to meet the goal confidence target is chosen.

Notes & Considerations

- For a “goal” allocation_priority, the minimal risk portfolio that satisfies the goal will be chosen, but the risk_score parameter is not considered explicitly. This may lead to an investor being allocated to a portfolio that is riskier than suitable for a given risk appetite.
- For a “risk” allocation_priority, risk appetite is the driving factor, and this may lead to a goal not being successfully achieved.

Goal Recommendation Appendix

Applicable Proton Services

- `POST /goal_accumulation/recommendation`
- `POST /goal_decumulation/recommendation`
- `POST /goal_accumulation/status`
- `POST /goal_decumulation/status`
- `POST /goal_accumulation/allocation`
- `POST /goal_decumulation/allocation`

Methodology

The process of generating a recommendation entails constructing a set of potential new goal plans by changing specific plan components and holding others constant. The engine determines a minimum viable outcome by searching through the universe of potential goal plans.

“Recurring” recommendations leverage the introduction of a new periodic recurring deposit, without altering the initial investment amount, the goal horizon, the portfolio attributes, or the target goal amount. An available domain is constructed based on the dep_min and dep_max parameters, and discrete values in the domain serve as the basis of a new goal plan universe. The smallest periodic deposit that projects to meet the goal confidence target is recommended.

“One-time” recommendations leverage the introduction of a new one-time deposit, without altering the periodic deposit amount, the goal horizon, the portfolio attributes, or the target goal amount. One-time deposits are attributed at time zero, effectively increasing the initial investment amount component of the goal plan. An available domain is constructed based on the inv_min and inv_max parameters, and discrete values in the

domain serve as the basis for a new goal plan universe. The smallest one-time deposit that projects to meet the goal confidence target is recommended.

“Combo” recommendations leverage the introduction of a new one-time deposit as well as a new periodic deposit, without altering the goal horizon, the portfolio attributes, or the target goal amount. One-time deposits are attributed at time zero, effectively increasing the initial investment amount component of the goal plan. An available domain is constructed based on the dep_min, dep_max, inv_min, and inv_max parameters. The permutation of a one-time deposit and recurring deposit with the smallest present value that projects to meet the goal confidence target is recommended.

“Horizon” recommendations leverage the introduction of a new accumulation horizon, without altering the initial investment amount, the periodic deposit amount, the portfolio attributes, or the target goal amount. An available domain is constructed by incrementing the original goal accumulation horizon forward, up to a maximum of 32 years above the original value. The shortest accumulation horizon that projects to meet the goal confidence target is recommended.

Notes & Considerations

- “Combo” recommendations, the domain of potential goal plans consists of all permutations of the potential one-time deposits and the potential recurring deposits.

Goal Tracking Appendix

Applicable Proton Services

- `POST /goal_accumulation/status`
- `POST /goal_decumulation/status`

Methodology

The current goal plan configuration is first used to execute a Monte Carlo simulation (see Monte Carlo Appendix section for more information). The result determines whether or not the goal is currently on track. For accumulation goals, in which the goal is defined by a total accumulation amount at the end of the accumulation horizon, final outcomes from the simulations must be greater than or equal to the target goal amount in order to be considered successful. For decumulation goals, in which the goal is defined by a periodic withdrawal amount over the course of the decumulation horizon, final outcomes from the simulations must be greater than zero in order to be considered successful. The goal probability (p) is computed as:

p = number of successes / number of simulations

The boolean tracking status is reported as true if p is greater than or equal to the user-defined confidence target, and reported as false if p is less than the user-defined confidence target.

Goal achievement scores are calculated by comparing p to the confidence target:

goal achievement score = p / confidence target

Goal progress is determined by comparing the current amount in the goal configuration to the target goal amount. For accumulation goals, this is defined as:

goal progress = current investment amount / target accumulation amount

For decumulation goals, a feasible decumulation amount is first estimated by lowering the target goal amount until the confidence target is satisfied. This answers the question: “If the decumulation horizon started today, what amount could be successfully withdrawn from the current balance over the desired decumulation horizon?” This value is used as the basis of the goal progress calculation, as follows:

goal progress = feasible decumulation amount / target decumulation amount

Notes & Considerations

- Goal achievement scores are normalized onto a 0-100 scale, where 0 indicates minimal likelihood of goal achievement and 100 coincides with an affirmative tracking status.
- If the recommend parameter is true, a recommendation will only be generated if the tracking status is negative.

Portfolio Construction

Mean-Variance Optimization Appendix

Applicable Proton Services

- `POST /mvo`
- `POST /risk_allocation`
- `POST /goal_accumulation/allocation`
- `POST /goal_decumulation/allocation`

Methodology

The model consumes an expected return for each security and the covariance relationships among all securities, which are derived directly from the security price data. Expected return is defined as the mean of a given security’s price distribution across the available data, while the covariance relationship between securities y and z is defined standardly as:

cov(y,z) = i=1n(y - y)(z - z)n - 1

The optimization process leverages a standard convex optimization protocol that solves problems via Second Order Cone Programming (SOCP)[i]. Problems are structured with three primary elements: an Objective, a

Variable, and Constraints. Here, the Objective is defined as a maximization of the portfolio mean return or a minimization of portfolio variance, the Variable is defined as an array of security weights, and the Constraints change based upon user input but include an upper or lower bound restriction on the portfolio's return or variance. Other user-defined constraints include:

- w_config - A series of weight constraints by security type:
  - Major securities are intended to be securities with more lenient weight thresholds.
  - Minor securities are intended to be securities with tighter weight thresholds.
  - Cash securities are constrained to a constant weight.
- w_asset_config - A series of weight constraints by security asset class:
  - Asset class constraints set a maximum weight threshold that applies to the cumulative sum of weights across securities of a given asset class.
- min_assets - The minimum number of securities to be included in an optimized portfolio (formally defined as the number of non-zero weights, excluding cash).

For efficient frontier optimization, a series of problems is solved in the following order:

1. A problem with an Objective to minimize portfolio variance.
   - This derives the portfolio with the lowest possible risk, which serves as the frontier domain's lower bound.
2. A problem with an Objective to maximize portfolio return.
   - This derives the portfolio with the highest possible return, which serves as the frontier domain's upper bound.
3. A problem with an Objective to maximize portfolio return for each risk value in the frontier domain, in ascending order.
   - Variance values are chosen along a logarithmic path between the variance derived from (1) and the variance derived from (2).
   - For each problem, the chosen variance is issued as an upper bound Constraint.
   - To maintain integrity of the frontier shape, new ascending solutions are only recorded if the optimal portfolio return is higher than previous portfolio returns.

For target portfolio optimization, a single problem is configured as follows:

- When targeting a risk value, the objective is set to maximize portfolio return and an upper bound constraint is placed on the portfolio variance.
- When targeting a return value, the objective is set to minimize portfolio variance and a lower bound constraint is placed on the portfolio return.

Notes & Considerations

- Problem convexity is enforced via Disciplined Convex Programming (DCP) criteria[ii].
- Minimum weight constraints are joined with zero to form a binary constraint. Hence, the weight w is allowable if $w = 0$ or $w \geq min$, but not if $0 < w < min$.
- An error will be raised if the problem cannot be solved. This commonly occurs in two cases:

  - When the security price data input into the model is incorrect or contains a very small amount of data points
  - When the user-defined constraints are too restrictive
- A maximum of 30 portfolios will be generated on the efficient frontier.

References

i. https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf
ii. http://dcp.stanford.edu/

Portfolio Management

Rebalancing Appendix

Applicable Proton Services

- `POST /rebalancing_signal`

Methodology

This service generates trade signals to strategically rebalance a model portfolio, with support for three types of rebalancing: period-based, drift-based, and Downside Protection. Period-based rebalancing generates signals at the beginning of each time period with predefined frequency. Drift-based rebalancing generates signals when a portfolio holding deviates from its strategic target by a predefined amount. Downside Protection generates signals based on the observed risk of portfolio holdings, driven by Hydrogen's proprietary risk management algorithm.

Hydrogen's Downside Protection algorithm has four main goals:

- Limit drawdowns[i]
- Reduce volatility and beta
- Track benchmarks on the upside
- Achieve superior risk-adjusted returns over the long-term

The primary driver of Downside Protection is dynamic asset allocation[ii], defined as a rules-based allocation strategy that includes responsiveness to market environments. Dynamic asset allocation differs from a passive or indexing strategy, which is static and not responsive (periodic or period-based rebalancing notwithstanding). Rebalancing within a dynamic framework is conditional rather than pre-determined. This also differs from an active strategy that uses forward-looking positions to try to take advantage of future market conditions. Dynamic allocations as described here are reactive and not predictive.

NAV III

Hydrogen's methodology utilizes a time-varying version of the CPPI[iii] framework (called TPPI[iv]). The time-varying portion is the level set for the floor, which changes based on market conditions and governs how rebalancing occurs. Based on quantitative risk measures looked at daily in the portfolio, the algorithm moves between risky and safe assets. The risky asset is weighted on a scale that ranges from 100% to 0% – meaning that the portfolio can be fully invested or fully protected at any given time, and can move up or down from each level in between. This allows the algorithm to ladder out of a position as it becomes riskier, and ladder in as it becomes less risky, in order to achieve an optimal mix for end-users.

The framework is extended for entire portfolios rather than for a single risky asset. Instead of looking at the portfolio as a whole, the algorithm assesses each holding individually to apply risk management, on a daily basis. This aims to achieve three objectives:

- **Set optimal risk thresholds for each asset class.** Commodities are much riskier than bonds, so different risk thresholds should apply.
- **Limit transactions and transaction costs.** By separating a portfolio into buckets, the algorithm avoids unnecessary transaction costs and maximizes the benefit of every trade.
- **Maintain portfolio integrity.** Each asset class has a strategic weight that acts as an anchor, so a given portfolio doesn't get riskier than intended (only less risky, when necessary) and no leverage is ever used.

Trades are triggered by the ongoing risk of each security in the model portfolio. Risk is assessed based upon the moving average current drawdown of a security and standardized onto a 0-100 scale. Downside trading occurs in 3 levels, each of which is triggered when the risk assessment value moves across predefined thresholds on the risk scale. Each level is associated with a given amount of exposure, which is defined on a relative basis as a percentage of the security's strategic weight. For example, a security may have levels of 75%, 50%, and 25% exposure triggered at 70, 80, and 90 risk assessment values. If the risk value moves from 69 to 71 for a given security, the security in question will be rebalanced to 75% of its original strategic weight. If the risk values drop from 71 back to 69, the security will be rebalanced to 100% of its strategic weight.

This framework can be applied to any type of portfolio – conservative or aggressive, diversified or concentrated, ETF-based or stock-based. Please note, using securities that have low liquidity may impede portfolio return through higher bid-ask spreads.

Notes & Considerations

- Each signal has an indicator for the applicable type of rebalancing:
  - 1 = period-based rebalancing signal
  - 2 = drift-based rebalancing signal
  - 4 = Downside Protection rebalancing signal
- One portfolio may have multiple rebalancing types activated.

References

i. "Optimal Investment Strategies for Controlling Drawdowns" published in Mathematical Finance, July 1993, by Dr. Sanford Grossman and Dr. Zhongquan Zhou, professors at The Wharton School, University of

Pennsylvania.
ii. “Dynamic Strategies for Asset Allocation” published in 1988 by Andre Perold of Harvard Business School and Nobel Prize winner William Sharpe.
iii. “CPPI – constant proportion portfolio insurance: protection with dynamic allocation control” published in 2006 by Martin Ermini from BNP Paribas. “Enhanced Balanced Portfolios With CPPI Methodologies” published in 2008 by Francesco Rossi of Pioneer Investments.
iv. “A Time-varying Proportion Portfolio Insurance Strategy Based on a CAViaR Approach” published in 2008 by Benjamin Hamidi, Bertrand Maillet, Jean-Luc Prigent, prominent financial academics in France.

Risk Scoring

Risk Score Appendix

Applicable Proton Services

- `POST /risk_score`

Methodology

Answer values are normalized onto a [0,100] scale based on their relationship to the corresponding max_answers values. These transformed answer values (x) contribute to the weighted average (wa) as:

wax = w1x1 + w2x2 … wnxn

Notes & Considerations

- Higher answer values and max_answers values are assumed to indicate more appetite for risk.
- Weights default to be equally weighted.

Dimensional Risk Score Appendix

Applicable Proton Services

- `POST /dimensional_risk_score`

Methodology

Answer values are first grouped by dimension, and then are normalized onto a [0,100] scale based on their relationship to the corresponding max_answers values. These transformed answer values (x) contribute to a

series of dimensional scores (ds) based on answer weights (aw) as:

dsx = aw1x1 + aw2x2 … awnxn

These dimensional scores are then combined to compute the total weighted score (tws) based on dimensional weights (dw) as:

tws = dw1ds1 + dw2 ds2 … dwndsn

Notes & Considerations

- The framework can handle any number of distinct dimensions.
- Individual answers may be mapped to one or multiple risk dimensions.
- When more than one question map to a given dimension, proportional relationships between various answer weights serve as the basis for weighting answers within that dimension.
- If a given dimension has no answers mapped to it, it is dropped from consideration and any remaining values are re-normalized.

Risk Allocation Appendix

Applicable Proton Services

- `POST /risk_allocation`
- `POST /goal_accumulation/allocation`
- `POST /goal_decumulation/allocation`

Methodology

The risk_score parameter, which is scaled from 0 to 100, is used to conduct a percentile analysis of the predefined portfolio risk values. When the target percentile falls between two portfolios, the risk value that is closer to the ideal percentile value on an absolute basis is chosen.

For the “create” allocation method, an efficient frontier domain is constructed based on the constraints defined in the opt_config parameter. The risk_score parameter, which is scaled from 0 to 100, is used to conduct a percentile analysis within the domain. A portfolio is then optimized, using the percentile value within the risk domain as a risk target (see the Mean-Variance Optimization Appendix section for more information).

Notes & Considerations

- The parameters `performance` and `volatility` must be defined in `/allocation` if “select” is chosen.
- For “select”, the portfolio with the closest match to the risk score will be selected.

NAV ☰

Simulations

Backtest Appendix

Applicable Proton Services

- `POST /rebalancing_signal`
- `POST /backtest`

Methodology

At the beginning of each period during the backtest period, the time series data for each security of the portfolio model is incremented up to the current date of the analysis. Weights are adjusted based on the periodic performance of each security in the portfolio, and asset size updates are computed with a simple weighted average of positional performance. Trades are determined via the model's rebalancing type, with the trading logic as follows:

For period-based rebalancing, the current date in the time series is analyzed on an absolute calendar basis. Annual period rebalancing to the model strategic weights is triggered if the date passes December 31st of a given year. Semiannual period rebalancing to the model strategic weights is triggered if the date passes June 30th of a given year. Quarterly period rebalancing to the model strategic weights is triggered if the date passes March 31st, June 30th, September 30th, or December 31st of a given year. Monthly period rebalancing to the model's strategic weights is triggered if the current date's month is greater than the previous date's month.

For drift-based rebalancing, rebalancing to the model's strategic weights is triggered if any security deviates from its strategic weight by a relative amount greater than its user-defined drift threshold. For example, a 10% drift factor applied to a security with a 20% strategic weight is triggered if the floating weight of the security falls below 18% or rises above 22%. When drift-based rebalancing is triggered, all securities in the portfolio are rebalanced to their respective strategic weights. For more information on downside protection, please see the Rebalancing Appendix section.

Notes & Considerations

- Drift thresholds are defined on a positional basis, and each portfolio security may have a unique drift factor.
- For Downside protection rebalancing, when the risk of a security triggers a trade, proceeds from any sale are shifted into a designated "safe" security. This safe security is assumed to be something with less risk than non-safe securities.

Monte Carlo Appendix

Applicable Proton Services

- `POST /monte_carlo`
- `POST /goal_accumulation/recommendation`
- `POST /goal_decumulation/recommendation`
- `POST /goal_accumulation/status`
- `POST /goal_decumulation/status`
- `POST /goal_accumulation/allocation`
- `POST /goal_decumulation/allocation`

Methodology

- For each period in the Monte Carlo simulation, the initial balance is applied at time zero. For each subsequent period, the following process occurs:
  - A return is sampled from a normal distribution with mu and sigma attributes defined by the user
  - The ret_mod value is applied to the sampled return to derive a modified return
  - The modified return is used to calculate a current balance from the previous balance
  - The applicable periodic cash flow is applied to the calculated balance to get the resulting balance for the period
- If remove_outliers is true, an outlier analysis is performed on a median absolute deviation (MAD) basis, at the standard 2.5 threshold